

RUN's  
10th Annual  
Special Issue  
Win a 128D

See P. 2

**FREE** Programmer's Reference Card Inside

# RUN

THE **COMMODORE** 64/128 USER'S GUIDE

1989  
An IDGC/I Publication  
Display until March 31  
U.S.A. \$3.95  
CANADA \$5.25

See  
Special  
Disk Offer  
P. 49

# Special Programming Issue

**Commands · Codes · Tutorials  
For Beginning & Intermediate Users**

**P L U S**

Type-in **ARCADE GAMES**

Never-Before-Published **MAGIC TIPS**

Answers to Your Questions  
About **COMMODORE** Computing



# Aussie JOKER POKER

# \$200,000 JACKPOT

The latest multi-player multi-format PC game from Australia is different to all forms of Poker.

**Aussie JOKER POKER** features 90 player capacity, open-ended discard ability, selectable deck size and hands per player, password controlled gambling system with automatic accounts — and **5 free entry forms for the \$200,000 Aussie JOKER POKER Contest.**

Each month December 1988 through April 1989 winners of 240 JOKER SOFTWARE games and 4 finalists will be randomly drawn from all entries received that month.

With a guest, the **20** Finalists will be flown to Las Vegas to play **Aussie JOKER POKER** for a **first prize of \$100,000 in cash** at the **Golden Nugget.**

## 1,220 Prizes Value \$200,000

1,200 Joker PC software games at \$29.95 to \$49.95 dependent on disk format. Game prizes at sole discretion of sponsor.

\$60,000

## Cash Prizes for Aussie JOKER POKER Contest Grand Final:

Highest Scorer: \$100,000

Second Highest Scorer: \$5,000

Third Highest Scorer: \$2,500

Lowest Scorer: \$1,500

16 Consolation Prizes of \$1,000 each to eliminated Finalists \$16,000

\$15,000

Prize includes air travel for Finalists and their guests from the major airport to Las Vegas with two days and two nights accommodation at the Golden Nugget (approx. retail value \$750 each subject to departure points).

All taxes and other expenses not specified herein are sole responsibility of winners. All winners will be notified in writing.



AMIGA™

## Aussie JOKER POKER is available for SIX major PC's

If your PC has a mouse or keyboard, a mono or color monitor and a 512K minimum ram (except Apple II and C64/128 use 64K and keyboard only) you and your family can practise at home for the Las Vegas final of the **Aussie JOKER POKER** contest.

Suggested retail prices:

**IBM & compatibles (CGA Board required) \$39.95**

**Amiga & Atari ST \$49.95**

**Macintosh (mono only) \$49.95**

**Apple II \$39.95**

**C64/128 \$29.95**

If ordering by telephone add \$3 shipping & handling and check that your PC meets the minimum hardware requirements as no cash refunds apply. Warranty is limited to free replacement of faulty products returned by prepaid post.

## Aussie JOKER POKER Contest Rules

1. No purchase necessary to enter.
2. Void where prohibited by state or federal law.
3. To enter, simply complete and return the the official entry form.
4. Limit five entries per family or household. Five free entry forms and full contest rules are included with "Aussie Joker Poker" or may be obtained by sending a stamped self-addressed envelope larger than 5 1/2" x 7 1/2" with a hand written request to: Aussie Joker Poker Contest Entry Forms, P.O. Box 22381, Gilroy, CA 95021-2381. Mail-in requests limited to one per name, household or family and must be received no later than 3/31/89. WA & VT residents need not include return postage. Full rules also available from participating Mindscape retailers.
5. Monthly entries must be received no later than the last day of the month in which a drawing will take place in order to participate in the month's drawing. Drawings will be held from December, 1988 through April 1989, inclusive. Final entries must be received by 4/30/89.
6. Contest open to legal residents of the U.S.A. and Canada (other than Quebec).
7. Odds of winning depend on number of eligible entries received.
8. Contest subject to complete official rules.

## SEE YOUR NEAREST MINDSCAPE SOFTWARE RETAILER



or if not available order direct on  
**1-800-24-JOKER**

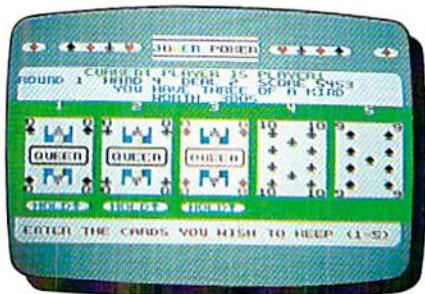
IN CANADA: 1-800-54-JOKER

24 hour order service

JOKER SOFTWARE INTERNATIONAL

PO BOX 22380, GILROY CA 95021-2380

RETAILERS CALL: 1-800-221-9884



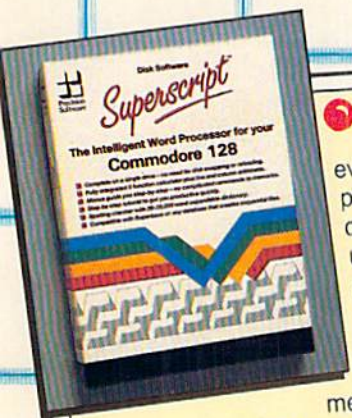
C64/128™



Another Wonder from Down Under B

IBM, Apple & Macintosh, Amiga, Atari ST and C64/128 are trademarks or registered trademarks of International Business Machines, Apple Computer, Inc., Commodore Amiga, Inc., Atari, Inc., and Commodore Electronics Ltd, respectively. © 1988 Joker Software JP31A

# The most powerful productivity software ever developed for your computer



SuperScript gives you everything you need for professional word processing in one easy to use package. Its menu command structure puts you immediately at ease with no complicated commands to memorize, yet SuperScript

combines business-style editing, spell checking, calculator, row and column arithmetic and full mail merge facilities. The phrase glossary feature enables you to store whole passages of text or Macro command sequences and recall them with a simple key.

Commodore 64 **\$34.95**  
Commodore 128 **\$39.95**

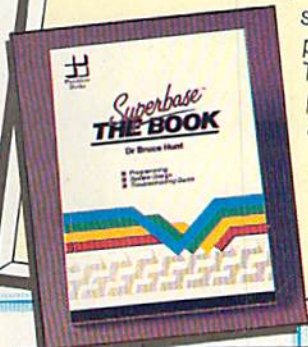
Buy both and get FREE "Superbase: The Book" A \$15.95 Value!

Superbase is the most powerful database system ever developed for 8 bit computers. Why? Because not only can you access its commands from menus but you can string them together with BASIC commands to form your own complete programs. Superbase can import data from and export to other programs via sequential files. In addition the C-128 version will load with, and pass Macro commands to, the SuperScript word processor to create a completely automated office system. **Now includes support for 1581 disk drive.**

Commodore 64 **\$39.95**  
Commodore 128 **\$49.95**



The in-depth guide to using the Superbase system, from first steps through to advanced programming techniques. The wealth of hints, tips and practical examples makes Superbase: The Book required reading for anyone working or contemplating working with Superbase. 194 pages **\$15.95**



SuperDiskdoc is the ultimate Commodore disk utility programme, and the best protection there is for your valuable data. Zoom in on the bytes on your disk, interpret them in hex, ASCII or plain English, make any changes you want, then replace them. SuperDiskdoc brings you extra security. In the event of an accident to your data you have the best tool available to repair the damage. Commodore **\$19.95**



**Attention Superbase Users!**  
Call us for information on 1581 upgrade and Official Superbase Information Network!

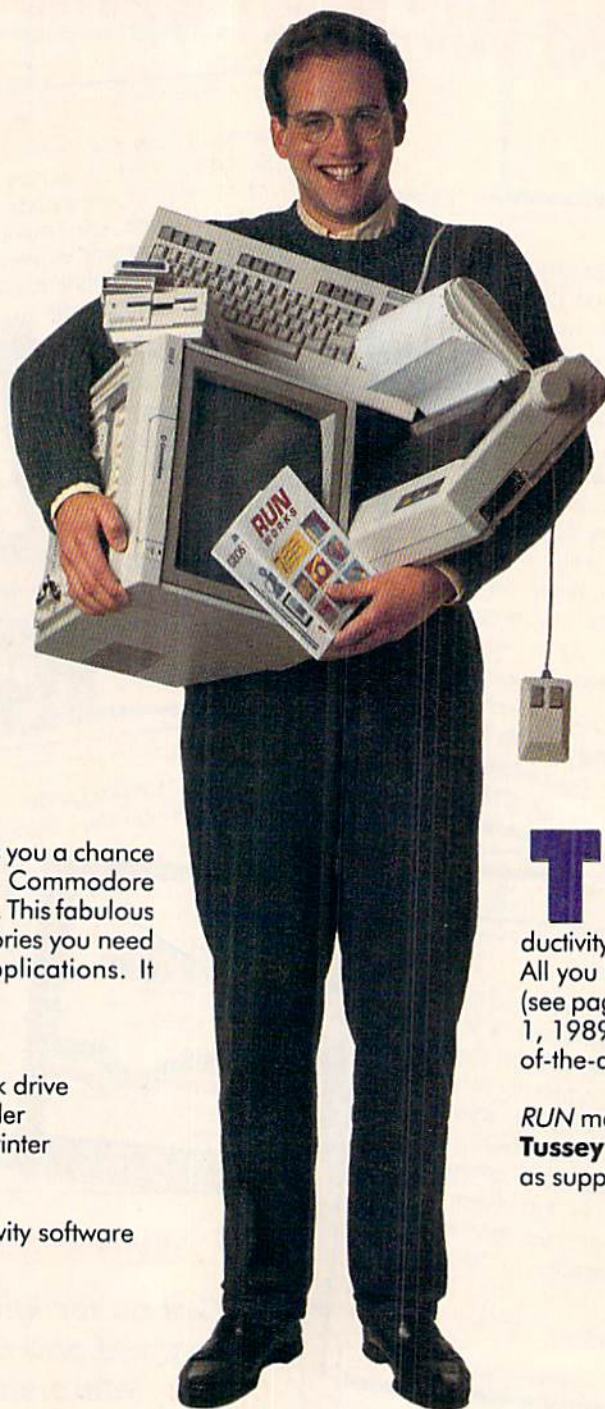
Available from your dealer or directly from -



Precision Incorporated,  
8404 Sterling Street, Suite A, Irving, TX 75063.  
Telephone: (214) 929-4888. Fax: (214) 929-1655.

Circle 419 on Reader Service card.

# You Could Win The Ultimate 128D System!



**R**UN magazine offers you a chance to win the latest in Commodore eight-bit technology. This fabulous prize contains all the accessories you need for all your computing applications. It includes:

- 128D computer
- 1084 monitor
- 1581 3½-inch disk drive
- 1750 RAM expander
- NX1000C color printer
- 1351 mouse
- 1670 modem

Plus: a collection of productivity software

**T**his computer prize (worth over \$1400) comes complete with computer, peripherals and productivity software. It could be yours for free! All you have to do is fill out the entry form (see page 97). Send in your entry by March 1, 1989, for your chance to win this state-of-the-art computer system!

*RUN* magazine extends its appreciation to **Tussey Computer Products**, who served as supplier of the hardware prizes.

To enter, simply fill out the entry form or a facsimile on page 97. All entries must be postmarked by March 1, 1989. Only one entry per household. The odds of winning will depend on the number of entries received. Taxes and duties on the prize are the sole responsibility of the winner. No substitutions will be made for any of the prizes. The prize is guaranteed to be awarded. All federal, state and local laws apply. Void wherever prohibited by law. Open to residents of U.S., its possessions,

Canada and Mexico. Anyone of any age may enter, but if won by a minor, the prize must be claimed by parent or legal guardian. Employees of IDG/Peterborough, its affiliates, subsidiaries, advertising and promotion agencies and the families of each are not eligible to enter. The winner will be selected in a random drawing held on March 31, 1989. *RUN* magazine will not be held responsible for lost, misdirected or late mail.

## FEATURES

### 6 MAGIC by Tim Walsh

Some 50 brand new Magic tricks to entertain and instruct you on your C-64 or C-128.

- 1. C-64 PROGRAMMING . . . . .6
- 2. C-128 PROGRAMMING . . . . .8
- 3. C-64 AND C-128 PROGRAMMING . . . . .10
- 4. APPLICATION PROGRAMS . . . . .13
- 5. DISK DRIVES . . . . .90
- 6. GENERAL HINTS AND TIPS . . . . .92

### 20 PROGRAMMING—Getting down to business.

#### 21 BASIC 101 by Annette Hinshaw

Get to know your computer better by introducing yourself to the Basic language.

#### 24 THE SECRET OF BETTER PROGRAMMING By John Ryan

Writing a computer program is an art. Follow these guidelines to help smooth the way.

#### 27 C-64 SPRITE BASIC by Charles Orcutt

Use these sprite commands to extend your command of sprites.

#### 31 C-128 SPRITE ACTION by Rob Kennedy

Master the use of sprites with these high-level Basic 7.0 commands.

#### 36 THE SOUND OF BASIC by Bruce Jaeger

Your Commodore is alive with the sound of Basic.

#### 44 IT'S ALL RELATIVE by Rob Kennedy

Basic 7.0 includes many new commands that make relative files easy to use.

#### 48 EXCUSE THE INTERRUPTION by Jim Hosek

Take full advantage of interrupts with these advanced machine language programming techniques. A three-part series.

### 64 GAMES—Time to break for some fun.

#### 65 TAG 'EM by John Fedor

Chase and get chased by your opponent in a mad dash around the C-64 court.

#### 68 GRAVITRON by Charles Orcutt

Hostile spaceships are plotting to steal Earth's precious water. Can you stop them or are you all wet? For the C-64.

#### 75 ASTRO-SHOOT by John Fedor

How many targets can you hit in 60 seconds or less? A C-64 arcade-action game.

### 78 COMMODORE CLINIC by Lou Wallace

Answers to your most-asked computing questions.

## DEPARTMENTS

### 2 CONTEST

Here's your chance to win a complete C-128D system, worth over \$1400.

### 4 RUNNING RUMINATIONS

The ingredients of this *RUN Special Issue* make for a tasty publication.

### 47 TYPE-IN TROUBLES?

Troubleshooting tips for entering listings.

### PROGRAMMERS REFERENCE CHART—at page 48

Twelve panels chock full of programming commands for the C-64 and C-128.

### 49 SPECIAL ISSUE DISK

Don't want to type in this issue's listings? Order this disk, which includes three unpublished bonus programs.

### 63 LEARN TO WALK BEFORE YOU RUN

First steps for new Commodore owners.

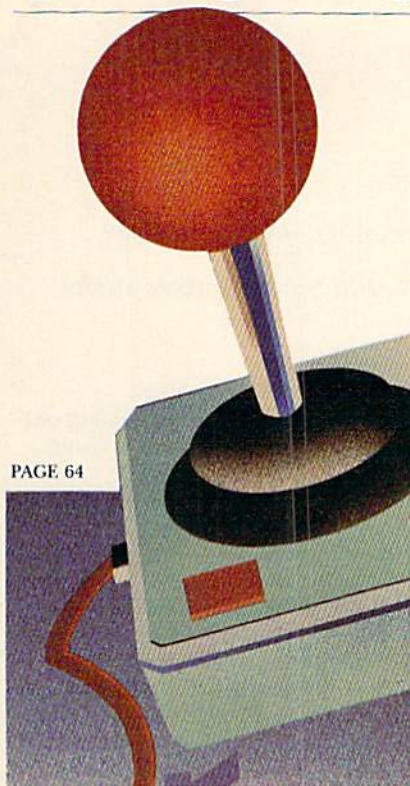
### 95 RUN'S CHECKSUM PROGRAM

Type-in help so you can RUN it right the first time.

### 96 AUTHORS WANTED!

Have you an interesting program or an article about Commodore computing? We'd like to see it!

### 96 LIST OF ADVERTISERS



PAGE 64



PAGE 7

**ABC** *RUN* (ISSN 0741-4283) is an independent journal not connected with Commodore Business Machines, Inc. *RUN* is published monthly by IDG Communications/Peterborough, Inc., 80 Elm St., Peterborough, NH 03458. Phone 603-924-9471. Second-class postage is paid at Peterborough, NH, and at additional mailing offices. Canadian second-class mail registration number is 9565. Subscription rates in U.S. are \$22.97 for one year, \$34.97 for two years and \$48.97 for three years. In Canada and Mexico, the one-year subscription rate is \$27.97, with U.S. funds drawn on a U.S. bank. Foreign surface mail subscriptions are \$42.97 for one year, and foreign air mail one-year subscriptions are \$77.97, with U.S. funds drawn on a U.S. bank. *RUN* is nationally distributed by International Circulation Distributors. Postmaster: Send address changes to *RUN*, Subscription Services, PO Box 58711, Boulder, CO 80522-8711. (Canadian address changes to *RUN*, PO Box 1051, Fort Erie, Ontario, Canada L2A 5N8.)

PUBLISHER  
STEPHEN ROBBINS

EDITOR-IN-CHIEF  
DENNIS BRISSON

MANAGING EDITOR  
SWAIN PRATT

SENIOR EDITOR  
BETH S. JALA

ASSOCIATE EDITOR  
HAROLD R. BJORNSEN

TECHNICAL MANAGER  
LOU WALLACE

TECHNICAL EDITOR  
TIMOTHY WALSH

COPY EDITOR  
PEG LePAGE

CONTRIBUTING EDITORS  
ROBERT KODADEK; ROBERT ROCKEFELLER;  
JOHN RYAN

ART DIRECTOR  
HOWARD G. HAPP

DESIGNERS  
ANNE DILLON  
LAURA JOHNSON

PRODUCTION  
ALANA KORDA

ASSOCIATE PUBLISHER AND  
NATIONAL ADVERTISING SALES MANAGER  
KENNETH BLAKEMAN

SALES REPRESENTATIVES  
NANCY POTTER-THOMPSON  
BARBARA HOY

CLASS AD SALES—EAST COAST  
HEATHER PAQUETTE  
603-924-9471

ADVERTISING COORDINATOR  
SUE DONOHUE

CUSTOMER SERVICE REPRESENTATIVE  
SUSAN MAIZEL

SECRETARY  
MARGOT SWANSON

WEST COAST OFFICE:

WESTERN STATES SALES MANAGER  
GIORGIO SALUTI

3350 W. BAYSHORE ROAD, SUITE 201  
PALO ALTO, CA 94303  
415-328-3470

CIRCULATION DIRECTOR  
PAUL RUESS  
1-800-525-0643

ASSISTANT CIRCULATION MANAGER  
PAM WILDER

MARKETING MANAGER  
WENDIE HAINES-MARRO

MARKETING COORDINATOR  
LAURA LIVINGSTON

EXECUTIVE ASSISTANT TO PUBLISHER  
LISA LAFLEUR

Entire contents copyright 1988 by IDG Communications/Peterborough, Inc. No part of this publication may be printed or otherwise reproduced without written permission from the publisher. Programs published in this magazine are for the personal use of the reader; they may not be copied or distributed. All rights reserved. *RUN* assumes no responsibility for errors or omissions in editorial or advertising content. *RUN* does not assume any liability for advertisers' claims.

# ***RUNNING RUMINATIONS***

*Our fifth annual Special Issue  
for Commodore 64 and 128 owners, programmers,  
non-programmers, experienced and novice users alike.*

Imagine that you're a magazine editor assigned the task of producing one issue to address the needs of Commodore users. What information would you include?

That's the question we posed as *RUN* editors assembled to formulate plans for this year's special issue.

"Let's do a programming issue," suggested one software techie, busily coding a program to cheat on his taxes.

"Hey, don't forget games," yelled the resident game junkie, carefully polishing his collection of joysticks.

One wily veteran chimed in, "Remember, our readers have always regarded the Magic and Commodore Clinic columns as their favorite features. We mustn't exclude either of them."

"Hey, how about a contest to give someone a chance to win a complete 128D system?" said one part-time staff member under the age of 18 and void where prohibited by law.

"Why not make the programs in the magazine available on disk so readers don't have to type in listings?" said the bleary-eyed proofreader, recovering from an overdose of machine language code. "And," she added, "we could offer some bonus programs, as well, to fill up the disk."

"Anything else?" asked the editor, furiously taking notes.

"I'll be sure to contact only the top authors and programmers in the industry to contribute," said the assignments editor, looking at his watch to determine the time in the South China Sea.

"Don't forget our programmer's wall chart, which is a big hit each year!" remarked one resourceful editor who had wallpapered her bathroom with charts from previous years.

"Aw, I don't have room on my walls for another poster," grumbled one worker. "Let's try something different—something readers will really remember this issue by."

"I've got it!" cried a voice of reason in the midst of impending chaos. "A folding programming card that users can tear out of the magazine and easily refer to while computing!"

"Those ideas are great," said the grizzled editor, "but I insist on one thing: This issue must contain something for everyone. Target the articles to address new computer owners, experienced computerists, novices, programmers, non-programmers, dabblers and dynamos. There must be something here for everyone, no matter what their level of expertise."

So here you have it—our fifth annual Special Issue, full of fun and solid C-64 and C-128 information that you'll want to refer to throughout the year.

You'll learn some fundamental programming techniques and, at the same time, improve your programming skills. Answers to your Commodore computing questions. Never-before-published Magic tricks. Blockbuster programs, games, applications, utilities, tutorials. Plus, a chance to win a complete 128D computer system worth over \$1400. And, the *piece de resistance*, our unique programmer's reference card that you can just tear out of the magazine and easily carry around in your shirt pocket.

For those who are familiar with our previous issues, you can expect more of the same quality editorial. For those who are new to *RUN*, welcome! You're in for a treat.

**Dennis Brisson**  
Editor-in-Chief

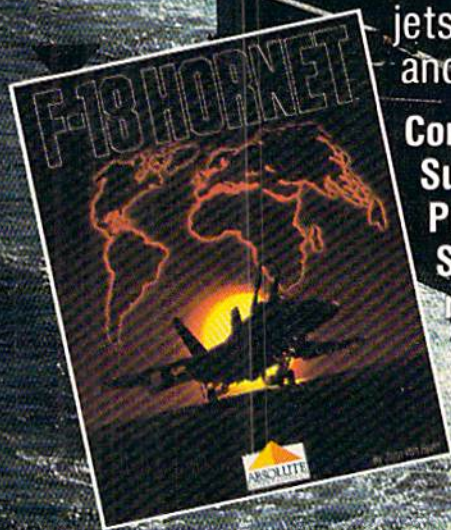
# ACTIVE DUTY!



F-18 HORNET™ is a carrier based adventure. Fast solid 3-D graphics and responsive instrumentation make for an incredible sensation of flight.

Your tour of duty will take you around the world on some of the most challenging and dangerous missions of your career. Fly through a variety of terrain executing aerial combat, emergency supply drops, strafing and bombing runs while battling hostile jets, helicopters and tanks.

Commodore 64™  
Suggested Retail  
Price: \$34.95  
See your  
retailer or call  
1-800-227-6900  
to order direct.



Commodore 64™ screens shown



F-18™ HORNET™ is a trademark and ABSOLUTE ENTERTAINMENT™ is a registered trademark of ABSOLUTE ENTERTAINMENT, INC. Distributed by MEDIA GENIC, Manufactured by ABSOLUTE ENTERTAINMENT, INC., PO Box 116, Glen Rock, NJ 07462  
© 1988 ABSOLUTE ENTERTAINMENT, INC. All Rights Reserved.

Circle 405 on Reader Service card

# Magic

*This is our fifth Special Issue Magic, containing some 50 Magic tricks. Unlike most of the previous Special Issues, where we included the best tricks from the year's regular issues of RUN, this time all the tricks are brand new. So warm up your machine and bring some magic to the new year.*



Compiled by TIM WALSH

LET'S TALK ABOUT the weather a moment. The summer of '88 will be remembered by New Englanders as the Thunderstorm Summer, with one storm following another from late June to late August. The frequent lightning also makes sitting at the computer and typing in Magic tricks a thrilling but dangerous sport.

As a storm approaches, most folks turn off their computer, television and other electrical devices, and if you don't, you should. Moreover, although you may keep your computer systems plugged into power strips equipped with on/off switches, even the best surge-protectors don't guarantee them immunity from lightning damage unless they're unplugged from the wall outlet.

Oh-oh. I see white flashes through the windows, so I must quickly save what I've written so far, and then unplug my system. I hope *RUN* readers will heed my hint when they read this Special Issue months from now.

—TIM WALSH, MAGIC COLUMNIST

## 1. C-64 PROGRAMMING

### C-64 SIGHT SAVER

The C-64's 16 colors look great, but getting a favorite screen, border and character-color combination involves a time-consuming process of finding the right Poke commands and keypresses. That's why I wrote C-64 Sight Saver. Just load and run it before a programming session. You'll then be able to change the background, border and cursor colors at will by holding down the F1, F3 and F5 keys, respectively. A built-in delay allows about one second between color changes.

```
Ø REM C-64 SIGHT SAVER - MARC TEMANSON :REM*25Ø
1Ø FOR X=49152 TO 49223:READA:POKEX,A:NEXT :REM*48
  :SYS 49152:NEW :REM*48
2Ø DATA 12Ø,169,192,141,21,3,169,13,141,2Ø :REM*11
  ,3,88,96,165,162,2Ø1,Ø,24Ø,15
3Ø DATA 2Ø1,64,24Ø,11,2Ø1,128,24Ø,7,2Ø1,19 :REM*1Ø3
  2,24Ø,3,76,49,234,165,2Ø3,2Ø1,4
4Ø DATA 24Ø,11,2Ø1,5,24Ø,13,2Ø1,6,2Ø8,21,7 :REM*32
  6,63,192,238,33,2Ø8,76,49,234
```

```
5Ø DATA 238,32,2Ø8,76,49,234,238,134,2,76, :REM*232
  49,234,76,49,234
```

—MARC TEMANSON, PEABODY, KS

### CLEARING 64 GRAPHIC AREAS

I wrote this versatile machine language routine to rapidly clear the C-64's hi-res screen in less than one second. Use it as a routine to clear the hi-res screen between loads. After running the program, poke the page number of the bit-mapped screen currently in memory in location 49153. The page number is determined by dividing the starting address by 256. For example, a screen with a starting address of 7168 requires the following command to be entered in Direct mode:

```
POKE 49153,28
```

If a page is not set, the program assigns a default page 32 address (memory location 8192).

```
Ø REM 64 HI-SPEED HI-RES SCREEN CLEARER - :REM*39
  SAULO DE LUCENA COELHO
1Ø FOR I= 49152 TO 49197:READ A:B=B+A:POKE :REM*81
  I,A:NEXT
2Ø IF B <> 7171 THEN PRINT "ERROR IN DATA :REM*14Ø
  STATEMENTS":END
3Ø DATA 169,32,133,252,169,Ø,133,251,24,16 :REM*178
  5,252,1Ø5,31,133,254,16Ø
4Ø DATA Ø,152,145,251,24,165,251,1Ø5,1,133 :REM*237
  ,251,165,252,1Ø5,Ø,133
5Ø DATA 252,165,251,2Ø1,64,2Ø8,234,165,252 :REM*222
  ,197,254,2Ø8,228,96
```

—SAULO DE LUCENA COELHO, BETHESDA, MD

### 64 HI-RES MADE EASY

High-resolution graphics performed from Basic on the C-64 is a concentrated study in peeking, poking, setting and clearing that tends to discourage all but the most ambitious programmers. Furthermore, unless you use a hi-res machine language utility, such as the above trick, setting up a hi-res screen from Basic is a slow process, requiring about 30 seconds. ▶





GA

## MAGIC

I wrote Hi-Res Set-Up and Clear for the C-64 to make the process more user-friendly and much faster. In fact, the first (main) program, like the above trick, clears a hi-res screen on the C-64 in less than one second, and also sets it up for use with other graphics programs. It can be appended to an existing Basic program or used as a stand-alone.

The second program is a short Basic program that shows when and how to activate the main program with a SYS 49152 and then plot a sine wave. Users can change the foreground and background colors of the hi-res screen, both of which share memory location 49205, which is currently a 16 for a black foreground and a white background. Try experimenting with various colors by poking different values into that location. You're certain to find uses for Hi-Res mode now that accessing it is much easier.

```
Ø REM 64 HI-RES SETUP PROGRAM - MIKE CORRIGAN
Ø REM*77
1Ø T=Ø :REM*8Ø
2Ø FOR X=49152 TO 49216:READ A:POKE X,A:T=
T+A:NEXT :REM*115
3Ø IF T <> 898Ø THEN PRINT"ERROR IN DATA ST
ATEMENTS...":END :REM*175
4Ø DATA 169,Ø,133,251,169,32,133,252,162,3
1,16Ø,Ø,169 :REM*237
5Ø DATA Ø,145,251,2ØØ,2Ø8,251,23Ø,252,2Ø2,
16,246,173 :REM*162
6Ø DATA 17,2Ø8,9,32,141,17,2Ø8,173,24,2Ø8,
9,8,141,24 :REM*2Ø9
7Ø DATA 2Ø8,169,Ø,133,251,169,4,133,252,16
2,3,16Ø,Ø,169 :REM*85
8Ø DATA 16,145,251,2ØØ,2Ø8,251,23Ø,252,2Ø2
,16,246,96 :REM*162
9Ø REM - HIRES DEMO PROGRAM #2 BELOW :REM*223
1ØØ POKE 5328Ø,Ø:REM BLACK BORDER :REM*32
11Ø SYS 49152:REM CLR SCREEN & ACTIVATE HI
-RES MODE :REM*23
12Ø S=8192:REM STARTING POINT ON SCREEN :REM*238
13Ø FORX= Ø TO 319 STEP .5 :REM*4
14Ø Y=INT(6Ø+5Ø*SIN(X/1Ø)):REM CALCULATE Y :REM*24Ø
15Ø C=INT(X/8):REM CHARACTER POSITION :REM*25
16Ø R=INT(Y/8):REM ROW :REM*213
17Ø L=Y AND 7:REM LINE :REM*122
18Ø B=S+R*32Ø+8*C+L:REM BYTE :REM*75
19Ø I=7-(X AND 7):REM BIT :REM*112
2ØØ POKE B,PEEK(B) OR (2{UP ARROW}I):REM S
ET BIT :REM*58
21Ø NEXT X :REM*6
22Ø GOTO 22Ø:REM FREEZE SCREEN :REM*1Ø9
```

—MIKE CORRIGAN, CARROLLTON, TX

### DOODLE! DISPLAYS MADE SIMPLE, TOO!

You can display Doodle! screens in your own C-64 programs and not have to worry about fancy programming, Peeks, Pokes or Graphics modes. Just load in any Doodle! screen in either Direct or Program mode with the command:

LOAD "DDFILENAME",8,1

Next, activate my program from either Direct or Program mode. Whenever you want to display the screen, place a SYS 51200 in either (you guessed it) Program or Direct mode, and the Doodle! screen will appear.

For added convenience, this routine is designed so that pressing the space bar returns the computer to the exact point in the program where you left off.

```
Ø REM C-64 DISPLAY DOODLE! - JEREMIAH MANN :REM*98
1Ø FOR T=512ØØ TO 5124Ø:READ D:POKE T,D:CK
=CK+D:NEXT :REM*92
2Ø IF CK <> 5Ø29 THEN PRINT"ERROR IN DATA S
TATEMENTS":END :REM*189
3Ø PRINT"OK, TO DISPLAY DOODLE!, ENTER SYS
512ØØ" :REM*18
4Ø DATA 169,59,141,17,2Ø8,169,12Ø,141,24,2
Ø8,169,198,141,Ø,221,32,228 :REM*166
5Ø DATA 255,2Ø1,32,24Ø,3,76,15,2ØØ,169,2Ø,
141,24,2Ø8,169,11,141,Ø,221 :REM*8
6Ø DATA 169,27,141,17,2Ø8,96 :REM*216
```

—JEREMIAH MANN, VISALIA, CA

## 2. C-128 PROGRAMMING

### FUNCTIONAL C-128 FUNCTION KEYS

Any time Commodore 128 users want to work on a program, they can load such files from disk more easily with my DLoad Key Assignment program.

The program, when run, prompts you to enter a number between 1 and 10. Numbers 1 through 8 program the corresponding function key to load a file. Number 9 lets shift/run-stop load a file, and 10 tells the help key to load a file.

Once you've run the program and assigned a key, list the directory, move the cursor up to the filename you want to load and press your designated load key. The word DLoad appears, the filetype PRG disappears and the program loads. Change the value of L\$ to "RUN" if you prefer to run a program rather than just load one.

```
Ø REM C-128 DLOAD KEY ASSIGNMENT - LEO BRENNEMAN :REM*66
1Ø PRINTCHR$(147)"ENTER 1 THRU 1Ø TO PROGR
AM" :REM*21
2Ø PRINT"A FUNCTION KEY, RUN/STOP OR HELP.
":INPUTA :REM*182
3Ø L$="DLOAD"+CHR$(34)+CHR$(27)+"O"+CHR$(2
7)+CHR$(75)+CHR$(2Ø)+CHR$(2Ø)+CHR$(2Ø)+
CHR$(13) :REM*1ØØ
4Ø L=LEN(L$):X=252:P=POINTER(L$) :REM*183
5Ø BANK1:POKEX,PEEK(P+1) :REM*5Ø
6Ø POKEX+1,PEEK(P+2):POKEX+2,1 :REM*164
7Ø BANK15:SYS 65381,X,A,L :REM*1Ø9
8Ø BANK1:POKEX,PEEK(P+1) :REM*84
9Ø POKE X+1,PEEK(P+2):POKEX+2,1 :REM*13Ø
```

—LEO BRENNEMAN, ERIE, PA

### RE-INITIALIZING C-128 FUNCTION KEYS

When you exit most C-128 programs, one or more function keys are either still re-defined to the program's definitions,

## MAGIC

or, more likely, they contain no definitions at all. You normally have three choices to restore the keys to their default C-128 definitions: Load a binary or Basic file of previously saved key definitions from disk, type in the definitions again, or reset the computer and lose the Basic program in memory. A fourth option is the best choice: Enter the following command in either Program or Direct mode:

```
BANK 15: SYS 49425: SYS 52526.
```

This is the same code used internally by the C-128 to install the default function key definitions on start-up or reset.

—MICHAEL MCGUIRE, COLORADO SPRINGS, CO

### GETTING C-128 HI-RES INTO DOODLE!

C-128 hi-res graphics screens can easily be loaded into Doodle! and Doodle!-compatible programs. Make sure you're in Graphics 1 mode and have a drawing on the hi-res screen. Then, hold down the run-stop key to halt program execution. Now, because you're in Graphics 1 mode, you can't see the text you're about to type, but don't worry. Just carefully enter the following statement:

```
BSAVE"DDFILENAME",B0,P7168 to P16192 <press return>
```

Your hi-res screen is now saved to disk in Doodle! format and can be loaded as a standard Doodle! screen.

—ALFREDO PADILLA, CUDAHY, CA

### BLOAD IT!

C-128 users with 1571 or 1581 drives have a wonderful speed advantage over 1541 users. Unfortunately, many great utilities, games, and applications work exclusively in 64 mode. Did you know that most machine language programs loaded in 128 mode will still be there once you switch to 64 mode? About the only exception are those containing zero page work areas and copy-protected commercial programs.

So, the next time you want to load in that long ML utility for the C-64, BLoad it in C-128 mode first, type GO64, then enter the SYS command to activate the program as you normally would. It'll be up and running in a fraction of the time.

—JOHN RYAN, BILOXI, MS

### RESCUING C-128 PROGRAMS

One of the most valuable tricks long-time C-128 users know is the program-rescue technique with the run-stop key. Should your C-128 lock up while running a program in 128 mode, don't despair. Hold down the run-stop key and press the reset button. Unless memory is corrupted by the lock-up, you'll start up in the C-128's machine language monitor. Press X and return, then list your program. Most of the time, it will still be intact.

—DOUGLAS JOHNSON, LARGO, FL

### SELECTIVE C-128 RUN-STOP DISABLE

If you write C-128 programs, you'll like using this simple but powerful command to enhance them. Just place these two lines at the beginning of your programs:

```
10 TRAP 15
15 IF ER = 30 THEN RESUME
```

By placing the Trap command in line 10, the computer branches to line 15 whenever it encounters an error. The ER = 30 in line 15 tells the computer when the run-stop key

is pressed, without disturbing the program's execution. The run-stop/restore key combination is unaffected.

—JOHN P. ROBINSON, JACKSON, MO

### MORE ON TRAPPING RUN-STOP

In the previous trick, the Trap command instructs the C-128 to detect errors, and it can also be a useful key press detector for the run-stop key. The following program shows how to configure the Trap statement so that when you press the run-stop key, it will abort a menu selection and send you back to the main menu. Try this technique in your next C-128 program.

```
Ø REM TRAPPING C128 RUN/STOP KEY PRESSES -
  JEROME E. REUTER :REM*186
5 TRAP 6Ø :REM*217
1Ø GOTO 3Ø :REM*136
2Ø PRINT"STOP KEY SENDS BACK TO MAIN MENU"
  :GOTO 5 :REM*14
3Ø PRINT"CHOOSE 1, 2 OR 3" :REM*76
4Ø DO:GETA$:LOOP UNTIL VAL(A$)>Ø AND VAL (
  A$) < 4 :REM*42
5Ø RUN :REM*188
6Ø CLOSE 4:GOTO 2Ø :REM*54
```

—JEROME E. REUTER, LADSON, SC

### 2 COMPARING FILES

Most Commodore computerists know that Basic programs residing in memory can be compared—from Direct mode—with files on disk by using the Basic 2.0 Verify and Basic 7.0 DVerify commands. If you'd like to try this command, type in a short program, such as:

```
10 PRINT"LET'S VERIFY THIS PROGRAM"
```

Save it to disk with the filename TEST. Immediately following the save operation enter, in Direct mode:

```
VERIFY"TEST",8 (Note: C-128 users can type: DVERIFY"TEST")
```

You'll get a message on the screen stating either OK or, if you made changes to the program after the save, ?VERIFY ERROR. While this process works fine with Basic programs, it won't work with binary (machine language) or program files created on word processors.

However, C-128 users, regardless of their programming skill levels, can use their computer's built-in machine language monitor to make effective binary comparisons of either two binary files or two word processor program files on disk. Here's how:

Load in the first file to compare into free memory in bank 0, using the command:

```
BLOAD"FILENAME",B0,P(DEC("1300"))
```

Immediately load the second file in bank 1, using the command:

```
BLOAD"FILENAME",B1,P(DEC("1300"))
```

Now press the F8 key to access the ML monitor, and find the end of the first file in bank 0 by entering:

```
M 1300<return>
```

Keep pressing M and return until you see an area of empty memory designated by a screenful of repeating numbers. Then write down the last memory location containing data ▶

## MAGIC

from your first file, which, for demonstration purposes, is 15CA. Now enter:

C 1300 15CA 11300

The first file in memory locations 1300 to 15CA is compared with the second file in bank 1, location 1300 (bank 1 memory addresses are prefixed with a 1). Any memory location that does not correspond is listed on the screen. Practice a few times with this technique, and you'll wonder how you ever lived without your C-128's ML monitor.

—VIRGIL PETERSON, KITTELY, ME

### SQUEEZING MORE INTO HELP

Commodore 128 programmers can redefine the help key as a function key. For example, the following two program lines reconfigure the help key to switch the 1571 disk drive to 1541 mode. Accessing the help key is performed via the BANK15: SYS 24812,,9,, command in line 20.

```
10 A$=CHR$(34)
20 BANK 15:SYS24812,,9,, "OPEN15,8,15," + A$ + "U0>M0" +
  A$ + ":CLOSE15" + CHR$(13)
```

—PETE LOWAS, HOT SPRINGS, AR

### C-128 PIANO KEYPAD

Turn your C-128's numeric keypad into a piano with this short program. Use numbers 1 through 9 to play notes, the decimal point to insert rests and zero to replay your composition.

```
Ø REM KEYPAD PIANO - ANDREW WATZNAUER
:REM*46
1Ø SCNCLR:PRINT"{2 CRSR DNs} 7:G 8:A 9:B
:REM*11Ø
2Ø PRINT" 4:C 5:D 6:E :REM*185
3Ø PRINT" 1:F 2:G 3:A :REM*127
4Ø PRINT" Ø:PLAY .:REST :REM*71
5Ø DIMA(2ØØ):FORN1= 1 TO 9 :REM*1Ø1
6Ø READN2$:N$(N1)=N2$ :REM*87
7Ø NEXT :REM*2ØØ
8Ø DATA "Ø4QF","Ø4QG","Ø4QA","Ø4QC","Ø4QD"
  ,"Ø4QE","Ø3QG","Ø3QA","Ø3QB" :REM*131
9Ø GETKEYA :REM*179
1ØØ IF PEEK(212)=82THEN N$(A)="R":GOTO 12Ø
:REM*176
11Ø IFA=Ø THEN PLAY N3$ :REM*84
12Ø PLAY N$(A) :REM*99
13Ø N3$=N3$+N$(A) :REM*117
14Ø GOTO9Ø :REM*21
```

—ANDREW WATZNAUER, NEW EGYPT, NJ

### C-128 HI-RES TEXT MODE

Commodore 128 owners are going to be in for a treat when they see this wild trick. It displays a hi-res image and lo-res text simultaneously on the 40-column screen. Type in the program, using RUN's Checksum, save it and then run it to see an example of this technique in action.

When the demo is finished, delete line 40 and save it to disk under a different filename.

Now you can enter graphics commands and watch the effect instantly, without the usual split-screen. For example, at the ready prompt, enter: BOX 1,0,0,319,199.

The program works by toggling between Hi-Res mode and Text mode 60 times per second to give the illusion of both Text and Hi-Res mode sharing the same screen.

```
Ø REM C-128 GRAPHICS & TEXT - MARCO A. GON
  ZALEZ HAGELSIEB :REM*157
1Ø FOR I=2816 TO 2837:READD:POKE I,D:NEXT
:REM*1Ø6
2Ø COLORØ,7:COLOR4,7:COLOR1,15:GRAPHIC1,1:
  GRAPHICØ:SCNCLR:PRINT"TEXT & GRAPHICS":
  CHARØ,Ø,2Ø :REM*137
3Ø SYS 2816 :REM*153
4Ø FORI=5 TO 9ØSTEP 5:CIRCLE 1,16Ø,1ØØ,I,I
  /2,Ø,36Ø,I*2,1Ø:NEXT:REM DEMO LINE!
:REM*244
5Ø DATA 12Ø,169,13,141,2Ø,3,169,11,141,21,
  3,88,96,169,1,69,216,133,216,76,1Ø1,25Ø
:REM*98
```

—MARCO A. GONZALEZ HAGELSIEB  
GUADALAJARA, JALISCO, MEXICO

### C-128 WINNING FANFARE

Commodore 128 users might want to add this winning fanfare to their Basic 7.0 programs to signal victory after winning or attaining a high score.

```
Ø REM C-128 WINNING FANFARE - J.R. CHARNET
  SKI :REM*42
2Ø TEMPO25:WS$="V1O5T9QCFAO6HCO5QAO6HC"
:REM*94
3Ø PLAY WS$ :REM*4'
```

—JOSEPH CHARNETSKI, PLAINS, PA

## 3. C-64 AND C-128 PROGRAMMING

### C-64/128 CURSOR WITH GET OR GETKEY

Most budding C-64 programmers know how to flash the cursor with the Get command, but few know the techniques for flashing a cursor with the Get and Getkey commands on the C-128.

My program flashes the cursor in 64 mode and in both the 40- and 80-Column modes on the C-128. When you first run the program, it determines which computer it's operating in, then performs a Poke for a flashing cursor. Poke 204,0 makes a cursor flash in 64 mode, while Poke 2599,0 uses both the Get and Getkey commands to flash a cursor in the C-128's 40-Column mode. SYS 52591 also uses Get and Getkey to flash a cursor in the C-128's 80-Column mode.

```
Ø REM CURSOR WITH GET & GETKEY - LEO W. BR
  ENNEMAN :REM*91
1Ø IF PEEK(4Ø96Ø)THEN A=2Ø4:GOTO 4Ø:REM 64
  MODE :REM*253
2Ø IF RGR(.5)=Ø THEN A=2599:REM C128 4Ø-CO
  LUMN MODE :REM*129
3Ø IF RGR(.5)<>Ø THEN 5Ø:REM C128 8Ø-COLUM
  N MODE :REM*179
4Ø POKE A,Ø:GOTO 6Ø :REM*222
5Ø SYS 52591:REM 8Ø-COLUMN MODE ONLY
:REM*191
```

## MAGIC

```
60 PRINT"HERE'S GET WITH A CURSOR":REM*233
70 GETAS:IFA$=""THEN 70 :REM*33
```

—LEO W. BRENNEMAN, ERIE, PA

## IMPROVING YOUR INPUT

Anyone who's ever tried to place commas or colons in the text of a C-64 or C-128 Input statement knows the all-too-familiar Extra Ignored error message. That's because everything to the right of the comma or colon is truncated. Let's look at this problem by entering these three lines:

```
10 INPUT A$
20 PRINT "YOU ENTERED:"
30 PRINT A$
```

Now type in RUN. At the question-mark prompt, enter the following: A,B,C:D and press return. The following message will appear on the screen:

```
?EXTRA IGNORED YOU ENTERED: A
```

Now it's time for a little magic. Run the program again, except, at the question-mark prompt, enter "A,B,C:D". Using double quotes with Input statements means no more Extra Ignored messages!

—STEVEN E. WEIGAND, WEST CHESTER, PA

## CATCH-ALL RESET

Finally—a reset command for any Commodore 8-bit! No longer will you have to use 64 or 128 reset commands in your programs. By entering the following command in either Direct or Program mode, memory is cleared of Basic programs and the computer's start-up screen will appear:

```
SYS PEEK(65532)+256 * PEEK(65533)
```

To see the reset value of your computer, substitute the word PRINT for the word SYS in the above command.

—ROBERT V. TAYLOR, LITTLE ROCK, AR

## VERTICAL TABBING MADE EASY

Though the C-64 and C-128 lack a Vertical Tab command, you can sometimes use Poke 214,X (X = screen line number minus one) for 64 mode, and, for 128 mode, Poke 235,X (X = screen line number minus one). For example, place the following command in your Basic program:

```
POKE 214,9:PRINT (or POKE 235,9:PRINT)
```

The cursor will appear at the beginning of line 10.

Bear in mind that while these are handy Pokes, they're not 100-percent reliable. Both the cursor's position at the time of execution and the design of your program can occasionally have an adverse effect on their operation. Also, remember that the Print command must follow both of these Pokes.

—E. STUART JOHNSON, ATHENS, AL

## NO MORE BLAND PRINT STATEMENTS!

Now you can get your C-64 and C-128 programs to print messages on the screen accompanied by all the clanging and clicking of those wonderful low-budget science fiction movie computers. Don't put it off any longer—add my program, Sci-Fi Telemessage to both your C-64 and C-128 programs and place your message in A\$. No doubt Buck Rogers would. ▶

# ATTENTION

## ALL COMMODORE 64/64C, COMMODORE 128/128D AND AMIGA OWNERS

A complete self-tutoring BASIC programming course is available that starts with turning your computer on, to programming just about anything you want! This course is currently used in both High School and Adult Evening Education classes and has also formed the basis of teacher literacy programs. Written by a teacher, who after having taught the course several times, has put together one of the finest programming courses available today. This complete course of over 220 pages is now available for the **COMMODORE 64/64C, COMMODORE 128/128D** and the **AMIGA 500/1000/2000** computers. This course will take you step by step through a discovery approach to programming and you can do it all in your leisure time! The lessons are filled with examples and easy to understand explanations as well as many programs for you to make up. At the end of each lesson is a test of the information presented. Furthermore, ALL answers are supplied to all the questions and programs, including the answers to the tests. Follow this course step by step, lesson by lesson, and turn yourself into a real programmer! You won't be disappointed! We will send this COMPLETE course to you at once for just \$21.95 plus \$3.00 for shipping and handling. **If you do not think that this is the best self-tutoring course you have yet come across, then just send the course back to us within 10 days of receipt for the FULL \$24.95 refund. That is our written guarantee.**

## FOLLOW-UP COURSE

Also available, a 200 page course exclusively on sequential and relative files using a unique approach for those with very limited file programming experience. Set up your own personal and business records! — same author — same guarantee — same cost. Fill in the coupon or send a facsimile.

NAME: \_\_\_\_\_ RU

ADDRESS: \_\_\_\_\_

CITY: \_\_\_\_\_

STATE/PROV: \_\_\_\_\_ CODE: \_\_\_\_\_

I desire the BASIC programming course

I desire the FOLLOW-UP course on file handling

The computer that the course is needed for:

COMMODORE 64/64C  COMMODORE 128/128D

AMIGA 500  AMIGA 1000  AMIGA 2000

Send cheque or money order (in the currency of your country) for just \$24.95 to:

### Brantford Educational Services

222 Portage Road

P.O. Box 1327

Lewiston, New York 14092

Fax: (519) 759-7882

6 Pioneer Place

or Brantford, Ontario

N3R 7G7

Telex: 061-81260

## MAGIC

```

Ø REM 64/128 SCI-FI TELEMESSAGE - RICHARD
  PENN                                :REM*186
1Ø INPUT"ENTER A MESSAGE";A$:REM OR USE '1
  Ø A$="SAMPLE TEXT"                  :REM*71
2Ø FOR L=54272 TO 54296:POKE L,Ø:NEXT
                                          :REM*2Ø
3Ø POKE 54296,15:POKE 54277,Ø:POKE 54278,2
  4Ø:POKE 54273,34:POKE 54272,75 :REM*35
4Ø POKE 54284,Ø:POKE 54285,24Ø:POKE 5428Ø,
  14:POKE 54279,24                    :REM*39
5Ø FORT=1 TO LEN(A$):PRINTMID$(A$,T,1)"(CO
  MD @){CRSR LF}";:X=X+1:IFX<2THEN 7Ø
                                          :REM*55
6Ø X=. :POKE 54276,17:POKE 54283,17:POKE 54
  276,16:POKE 54283,16                :REM*1Ø7
7Ø FORY=1TO2Ø:NEXT:NEXT:PRINT" " :REM*176

```

—RICHARD PENN, MONTREAL, QUEBEC, CANADA

### 64/128 DROP-DOWN MENUS

Programmers who've longed to add the sleek look of a professional drop-down menuing system to their C-64 or C-128 programs will now find their dream come true. Here's a drop-down menuing system that'll work in both 64 and 128 modes! Type it in, using *RUN's* Checksum program, save a copy, and then run it. It begins by printing a row of eight menu selections across the top of the screen. (Note: With slight modification, it could take full advantage of the width of the C-128's 80-column screen.)

Next, press a number from 1 to 8, and the corresponding menu selection drops down beneath the number chosen. Use the cursor keys to move between the options in the drop-down menu and select the option desired by pressing return. For demonstration purposes, the program then informs you of your selection. I'm sure you'll find this routine has almost unlimited potential in programs using menus.

```

Ø REM DROP-DOWN MENUS - J.R. CHARNETSKI
                                          :REM*82
1Ø POKE5328Ø,2:POKE53281,Ø:NS=24:DIMDM$(NS
  ):L$="{12 SPACES}"                  :REM*11
2Ø FOR I=1TONS:DM$(I)="OPTION "+CHR$(64+I)
  :NEXT:RV$="{CTRL 9}":HL$="{CTRL 2}":MC$
  ="{CTRL 7}"                          :REM*74
3Ø PRINTRV$HL$"{SHFT CLR}MENU: 1{2 SPACES}
  2{2 SPACES}3{2 SPACES}4{2 SPACES}5{2 SP
  ACES}6{2 SPACES}7{2 SPACES}8{11 SPACES}
  {CRSR LF}"CHR$(148)" ":GOTO 9Ø :REM*114
4Ø IF M THEN FOR J=1 TO 7:PRINTTAB(M+3)"(C
  RSR UP)"L$"{CRSR UP}":NEXT          :REM*18
5Ø M=X*3:N=M-2:S=N:FORJ=1TO7:PRINTTAB(M+3)
  RV$MC$L$:NEXT                        :REM*25
6Ø PRINT"(HOME){CRSR DN}":FORJ=NTOM:IFS=J
  THENPRINTTAB(M+5)RV$HL$DM$(J)MC$:GOTO8Ø
                                          :REM*228
7Ø PRINTTAB(M+5)RV$DM$(J)              :REM*25
8Ø PRINT:NEXT                           :REM*225
9Ø GETA$:IFA$="{CRSR DN}"ANDMTHENS=S+1:ON-
  (S<=M)GOTO6Ø:S=N:GOTO 6Ø           :REM*63
1ØØ IF A$>"Ø" AND A$<"9" THENX=VAL(A$):GOT
  O4Ø                                    :REM*123
11Ø IF A$<>CHR$(13)ORM=Ø GOTO 9Ø      :REM*76

```

```

12Ø REM USE ON S GOTO/GOSUB FOR BRANCHING
                                          :REM*211
13Ø PRINTHL$"{SHFT CLR}{CRSR DN}YOU SELECT
  ED "DM$(S)                            :REM*16

```

—JOSEPH CHARNETSKI, PLAINS, PA

### ILLEGAL QUANTITY ERROR TRAPPING

The next time you get an Illegal Quantity Error message in a program such as *RUN's* Checksum program, don't panic and rush to cancel your subscription because you think the program is full of bugs. Your computer is simply trying to tell you that you made a mistake in the numerical Data statements. Here's a sure-fire solution to the problem: Append the following six lines to the end of the program in question.

```

60000 AA=0
60001 AA=AA+1:READ AD
60002 IF INT(AB)<>AB THEN 60010
60003 IF AB<0 OR AB>255 THEN 60010
60004 GOTO 60001
60005 PRINT AA;AB:STOP

```

Once they're appended, enter *RUN* 60000, and this program begins checking your Data statements for missing commas, periods mistyped for commas and other errors. When it encounters a problem, it'll print both the number and the number's position (1,2,3, etc.) in the Data statements. List the program, make the correction, then enter *RUN* 60000 again to find more errors.

When *OUT OF DATA ERROR* appears on the screen—don't panic—it's merely a signal that all of the data elements have been checked. Delete lines 60000 through 60005 and save your program, which should then work well.

—JOHN WELLNER, PORT HUENEME, CA

### RESOLVING ILLEGAL QUANTITY ERRORS

The Illegal Quantity Error message is usually produced by the presence of typing errors in Data statements. Over the years, Commodore computerists have swapped solutions to the message as native Texans swap chili recipes. Case in point: *Magic Trick \$49C* (May 1988) showed C-64 users how to find the line containing the offending data number with the Direct mode command:

```
PRINT PEEK(63)+PEEK(64)*256
```

My addendum to this bit of Magic is a C-128 version of that command:

```
PRINT PEEK(65)+PEEK(66)*256
```

Like the C-64 version, it reveals the line number of the data statement in error. If neither of these tricks reveals the line causing the error, add the following two lines to the program in question:

```
1 READ A:IF A=INT(A) THEN 1
2 PRINT PEEK(63)+PEEK(64)*256
```

(Note: C-128 users must substitute `PRINT PEEK(65)+PEEK(66)*256` in line 2.)

This process detects any periods mistakenly typed between data numbers. Once you've changed any periods that don't belong in a listing, you'll eventually see an Out of Data error in 1 message, but that's all right. Just finish off by deleting

## MAGIC

lines 1 and 2, and you should have a working version of your program.

—HELEN ROTH, LOS ANGELES, CA

```
100 PRINT"FIVE":RETURN           :REM*49
110 PRINT"SIX":RETURN             :REM*249
```

—HELEN ROTH, LOS ANGELES, CA

### TOO MANY GOTOS AND/OR GOSUBS?

In most Basic programs, computed Gotos and Gosubs must be placed in one line of code to work properly. But what if you have so many that you can't get them to fit on a line? Here's an easy solution: Put the additional Gotos or Gosubs on a second line and tell the computer to skip the first line if a variable exceeds a certain value. My program clearly demonstrates how this procedure is accomplished:

```
0 REM EXTRA COMPUTED GOTO'S & GOSUBS - HEL
  EN ROTH                               :REM*38
10 INPUT"ENTER A NUMBER FROM 1 TO 6";N
                                         :REM*17
20 ON -(N>3) GOTO 70                     :REM*153
30 ON N GOTO 40,50,60                   :REM*87
40 PRINT"ONE":GOTO 10                   :REM*200
50 PRINT"TWO":GOTO 10                   :REM*58
60 PRINT"THREE":GOTO 10                 :REM*10
70 ON N-3 GOSUB 90,100,110              :REM*161
80 END                                   :REM*208
90 PRINT"FOUR":RETURN                   :REM*152
```

## 4. APPLICATION PROGRAMS

### 8 X 8 GRAPH PAPER

When you're designing custom characters using a font editor, you'll save yourself time and minimize frustration by designing your characters ahead of time on grids that match Commodore screen characters' 8 x 8 dot matrix. My program, 8 x 8 Graph Printer, allows virtually any printer to print a total of 48 8 x 8 graphs per 8 1/2 x 11-inch page.

```
0 REM 8 X 8 GRAPH PRINTER - J.R. CHARNETSK
  I                                     :REM*212
10 PRINTCHR$(147)"POSITION PAPER, PRESS A
  KEY TO PRINT"                         :REM*226
20 GETK$:IFK$="" THEN 20                :REM*3
30 NS=8:PRINTCHR$(147)"WORKING..." :REM*123
40 OPEN4,4:CMD4:PRINT                   :REM*255
50 FORA=1 TO NS:FORB=1 TO 8:FORC=1 TO 6
                                         :REM*183 ▶
```

# Make It A Merry Christmas for the Commodore user on your list

## The COMPLETE Lottery TRACKER and WHEELER™

The MOST COMPREHENSIVE Lottery Software Program on the Market Today for PICK-6 games is now available for Commodore 64/128! Look at ALL of these Features:

- Record Hundreds of Past Winning Lottery Numbers and Dates!
- Track as many State or International Lottery Games as you want! No Limit!
- Produce EXPERT Trend Charts to Identify Those HOT and DUE Numbers!
- Analyze Hits 4 ways: Bell Curves, Recency, Percentages, Frequencies, MORE!
- Produce STATISTICS for ALL Numbers You Play — No Randomizing Here!
- Select Numbers to Play 5 Different Ways! You Choose what YOU Like Best!
- Check Your Bets For WINNING Combinations! Records ALL Systems Played including BONUS NUMBER, where applicable.
- Print Charts, Statistics, Recorded Numbers and WHEELING SYSTEMS!
- We Include FREE Addresses and Phone Numbers (where available) of ALL State and International Lottery Commission Offices for Winning Number Lists.

### Includes 20 of the Worlds MOST Popular WHEELING SYSTEMS!

Use your computer to improve your odds HUNDREDS of TIMES!

#### Look At What Our Customers Have To Say:

"I Hit 54 CASH PRIZES the first 8 weeks with the help of your program! The Tracker and Wheeler IS the BEST lottery software program I have used overall... Over \$2100 ahead after ALL expenses!" B.C., El Paso, TX

"I won 4 cash prizes the first 2 times I used the Tracker and Wheeler!"

B.L.M., Wilmington, DL

"The COMPLETE Lottery TRACKER and WHEELER is SPECTACULAR!"

E.D., New York, NY

### Many, Many More Letters from CASH WINNERS on File!

No other lottery software package provides all of these features! When we say complete, WE MEAN COMPLETE. Easy to use MENU DRIVEN SCREENS. Printer and Color Monitor recommended but not required for use. All wheeling systems and program features now take only SECONDS to complete! You will LOVE this program in COLOR! Why pay UP TO \$150.00 for less?

### Don't Hesitate! Place your Order Now!

ONLY: \$34.95 Plus \$2.00 S&H

Now Sold in All 50 States and 17 Foreign Countries!

NEW

## The Daily Number Buster!™

You won't believe it until you see it. A COMPLETE Software Package for 3 & 4 digit DAILY NUMBER GAMES!

- Stores 100's of past winning 3 & 4 digit numbers and dates!
- Print Charts, Stats, Position Hits & more!
- Position Hit Chart displays HOT & DUE numbers by Drawn Winners!
- Choose from 4 bet methods!
- Every straight & combination bet and all BOXING BETS!
- Save your bets & review against winning numbers!
- Complete Odds explanation chart on the BUSTER DISK!
- ... and MUCH, MUCH MORE!

If you play the Daily Number Games you will quickly see the advantages (and REWARDS!) of working with your computer to analyze and find those WINNING 3 & 4 digit numbers!

Call or write for Your Copy Now! Only: \$34.95 Plus \$2.00 S&H

NEW

## The 50 System Lottery Wheeler Plus!™

- 50 NEW wheels PLUS the ability to add your own favorites to the system!
- Use WITHOUT the Lottery Tracker OR Link to the Tracker Data Base to extract the Hot & Due Numbers!

All GUARANTEED Winning Systems!

Introductory Price Only: \$24.95 Plus \$2.00 S&H



Now!

MC/Visa Call Toll Free 1-800-824-7888, Ext. 283

For Canadian Callers: 1-800-544-2600

Entertainment On-Line®, Inc. P.O. Box 553, Westboro, MA 01581  
The PREMIERE Lottery Software and Audio Products Company

MC, Visa and MO orders shipped within 1 week. Please allow 3 to 4 weeks for Personal Checks.  
MA Residents add 5% sales tax. Dealers Inquires a MUST!! © Copyright Entertainment-On-Line®, Inc., 1988

## M A G I C

```

6Ø PRINTSPC(1-(C>1)*4);           :REM*116
7Ø FORD=1 TO 8:PRINTCHR$(2Ø7);:NEXT :REM*186
                                     :REM*24Ø
8Ø PRINTCHR$(165);:NEXT           :REM*4
9Ø PRINTCHR$(8):PRINTCHR$(15);:NEXT :REM*146
                                     :REM*35
1ØØ FORE=Ø TO 5:PRINTSPC(1-(E>Ø)*5); :REM*35
                                     :REM*145
11Ø FORF=1 TO 8:PRINTCHR$(163);:NEXT:NEXT :REM*145
                                     :REM*199
12Ø FORG=1 TO 3:PRINT:NEXT:NEXT :REM*199
13Ø PRINT#4:CLOSE4:PRINT"{CRSR DN}ALL DONE
"                                     :REM*89
                                     —JOSEPH CHARNETSKI, PLAINS, PA
8Ø FORT=256 TO 514:PRINT#1,CHR$(2);:NEXT
                                     :REM*186
9Ø FORT=1 TO 5Ø:READX:PRINT#1,CHR$(X);:NEX
T                                     :REM*9Ø
1ØØ FORT=1 TO 16:PRINT#1,MID$(BN$,T,1);:NE
XT                                     :REM*73
11Ø CLOSE1:END                       :REM*26
12Ø DATA 169,Ø,32,144,255,169,2,166,186,16
Ø,1,32,186,255,169,16                 :REM*71
13Ø DATA 162,53,16Ø,2,32,189,255,162,255,1
6Ø,255,169,Ø,32,213                   :REM*186
14Ø DATA 255,134,45,132,46,169,Ø,133,122,1
69,8,133,123,32,96,166                :REM*32
15Ø DATA 76,174,167                   :REM*24

```

—LARRY E. SUTTER, STERLING HEIGHTS, MI

### C-64 SCREEN SWEEPER

If a C-64 program you're writing needs a screen-clear routine, try this unconventional one. It uses a reversed graphics character that almost instantly covers the screen from top to bottom, then reverses the process to reveal the cleared screen.

```

Ø REM 64 SCREEN SWEEP - J.R. CHARNETSKI
                                     :REM*31
1Ø CH$="{SHFT +}":SW$="{CTRL 9}" :REM*165
2Ø FORI=1 TO 4Ø:SW$=SW$+CH$:NEXT :REM*148
3Ø SYS 58726:FORI=1 TO 25:PRINTSW$;:NEXT
                                     :REM*64
4Ø FORJ=1 TO 12:SYS59626:NEXT:SYS58692
                                     :REM*27
5Ø REM RETURN                         :REM*5Ø

```

—JOSEPH CHARNETSKI, PLAINS, PA

### C-64 AUTO-RUN MADE EASY

Use the following program to make a two-block auto-run file that automatically runs your C-64 programs when it's loaded with the syntax LOAD"FILENAME",8,1. Type in this program using RUN's Checksum, save it to disk, then run it. You'll first be prompted to enter the name of the two-block boot file. This is the file you'll load with the above syntax in order to auto-run the desired file. Enter the boot filename, and the program automatically saves itself to disk. Next, you'll be prompted to enter the filename of an existing program on disk that you want to auto-run.

While following this sequence seems a bit confusing, rest assured that it is easy to perform. You'll be able to auto-run Basic programs in no time.

```

Ø REM 64 AUTORUN MADE EASY - LARRY E. SUTT
ER                                     :REM*165
1Ø FOR J=1 TO 5Ø:READX:CK=CK+X:NEXT:RESTOR
E                                     :REM*94
2Ø IF CK<>6221 THENPRINT"ERROR IN DATA":EN
D                                     :REM*152
3Ø INPUT"ENTER NAME OF BOOT FILE";BF$
                                     :REM*18Ø
4Ø INPUT"ENTER BOOT NAME";BN$       :REM*134
5Ø BN$=BN$+"*":FORJ=1TO15:BN$=BN$+"-":NEXT
                                     :REM*15
6Ø BN$=LEFT$(BN$,16)                 :REM*15
7Ø OPEN1,8,4,"Ø":"+BF$+",P,W":PRINT#1,CHR$(
Ø);CHR$(1);                           :REM*187

```

### A DIFFERENT TYPE OF AUTO-RUN

Some machine language (ML) programs not only contain Basic programs that need to be loaded and run, but also require you to enter a SYS xxxxx command to activate the ML code. A classic example is the commercial pinball arcade game, David's Midnight Magic. Not only does DMM require a Basic boot program, but you must enter SYS 49152 to complete the loading process.

Below is an example of a C-64 Basic boot program that can be used to streamline the process of booting ML programs from disk without requiring the user to enter SYS commands to activate the program:

```

10 PRINT CHR$(147):PRINT:PRINT
20 PRINT"LOAD" CHR$(34)"FILENAME" CHR$(34)"8,1":
REM ENTER ML FILENAME
30 PRINT:PRINT:PRINT:PRINT
40 PRINT"SYS xxxxx":REM ENTER REQUIRED SYS NUMBER
50 PRINT CHR$(19):POKE 198,2
60 POKE 631,13:POKE 632,13

```

After you type it in, save it to the disk containing the ML programs requiring SYS calls. Substitute the name of the ML file for the word FILENAME in line 20 and the SYS number for the xxxxx in line 40. This loader works by printing the commands on screen and activating them by automatically pressing return after each command is printed. Line 50 homes the cursor and tells the computer to expect two keypresses (up to 10 keypresses are possible). Line 60 delivers those keypresses (two returns).

—HELEN ROTH, LOS ANGELES, CA

### LINE LOCKER

Teachers and classroom instructors who routinely give tests on computers are just one example of Commodore users who occasionally need to make Basic program code hidden from users to prevent unauthorized examination or alteration. Line Lock lets you modify C-64 and C-128 programs so that the List command will display only the line numbers, not the Basic code itself.

Type in and save Line Lock, then run it. Place the disk containing the file to be locked into the disk drive and, at the prompt, enter the current filename, a comma, and then the new filename. The new filename is a "locked" version of the first file, which remains unscathed.

Continued on p. 90.



**NOW! Your IBM  
Tandy  
TRS-80  
Apple  
Commodore  
or  
Compatible is . . .**



**a COMPUTER that  
WRITES PROGRAMS  
FOR YOU  
for  
1/2 OFF\***

*\* Save 1/2 or More on this Special Limited Offer  
Limited Offer Good for 30 Days*



# SAVE! Over 1/2 OFF an AUTOMATIC PRO

for your IBM or Compatible, Tandy,

To Computer Users,

Now you can tell your computer what you want and your computer can write your programs for you in minutes to your custom design—easily and without requiring any programming background from you . . . with QUIKPRO + II.

## A Breakthrough In Micro Computer Technology

You know your computer is fantastically fast . . . once it knows what to do. Programs and software are what makes it happen. Every task your computer performs for you requires some kind of program. Until now, you could only get programs in just one of two ways: buy a canned package that many times doesn't meet your needs or hand over hundreds or thousands of dollars for a custom programming job. Now, you have a better choice . . .

## Programs Without Programming

Automatic programming is what it's all about. And, with QUIKPRO + II the Automatic Program Writer, your computer can actually write programs for you. You can quickly generate a new individual application program when you want it with QUIKPRO + II . Each program you create is a completely stand alone program that will run in the standard BASIC language you already have on your own computer.

Best of all, **you do not have to become a programmer** to use QUIKPRO + II. The QUIKPRO + II software becomes your personal programmer, waiting to do your work for you any time of the day or night you choose to use it.

## How To Get Over 1/2 Off

Like all successful software QUIKPRO + II was originally sold for well over \$100 per copy, and we have sold thousands. So why are we willing to let you buy at less than half price? . . . because our tests prove that at \$29.50 we sell over ten times as many. Obviously this cuts our costs because of the huge volume, so **we pass all savings directly to you.**

So, **ORDER Now.** Call Toll-Free 24 Hours or Mail in your Order. This offer is limited to those computer types listed on the Special Discount Order Form.

QUIKPRO + II comes complete in its own vinyl storage binder, with 80 page manual and disk ready to use.

## APPLICATION CHECKLIST

Here are a few of the thousands of possible applications you can do with QUIKPRO + II. . . And most can be created in a few minutes.

### BUSINESS USES

- Customer Filing
- Master Files for General Ledgers
- Accts. Receiv.
- Accts. Payable
- Telephone Logs
- Telephone Lists
- Hotel/Travel/Data Reservations
- Property Control
- Library Catalogues
- Inventories
- Key Employee Data

### EDUCATIONAL USES

- Student Records
- Grade Records
- Teacher Lists
- School Lists
- Program Design
- Course Design
- Tuition Data
- Enrollment Data
- Property/Equipment
- Athletic Schedules
- Player Statistics
- Test Scores
- Menus

### HOME & HOBBY USES

- Personal Records
- Check Lists
- Club Rosters
- Telephone Directories
- Recipe Files
- Medical Information
- Insurance Records
- Tax Records
- Christmas Gift Lists
- Deposit Files
- Due Dates
- Mortgage Data
- Travel Records

Not to mention the unlimited number of general filing, and crossfiling, technical and scientific uses.

# Get **QUIKPRO + II** **GRAM WRITER**

**TRS-80 or Apple Computers.**

## **The All-In-One Program**

The custom programs you can generate from the new QUIKPRO + II will let you perform Personal Filing, Fast Data Retrieval, including Changes, Deletions and Searches. You can selectively Print Custom Letters, all kinds of forms (if you have a printer). This new feature is called **Free Form Reporting**. You can even include calculations in the programs you create. QUIKPRO + II is perfect for creating inventory programs. You can use QUIKPRO + II to prepare letters and selectively address the letters to only certain people. And of course you can **SORT** your reports so that they print out information in the order that you want it, or print out only certain information. In fact, you can actually use QUIKPRO + II to create an easy to use Data management program or a simple spread sheet. You can do all of this and more with this All in One Program...and the best part is that **you need no BASIC programming experience.**

## **How Does It Work?**

You can do it simply by answering easy questions that appear on your screen. You won't have to learn any Computer commands or special Programming Languages. Instantly the QUIKPRO + II software instructs the computer to write efficient error free, BASIC Programs and puts the **Programs right onto your own disk, ready for you to use.**

The resulting custom program is truly a separate BASIC program. You can list it. You can modify it. You can customize it to your own liking. You can actually see what makes it tick.

## **What People Say About QP Software**

From a GENERAL CONTRACTOR/CONSTRUCTION COMPANY owner:

*"The program seems to be good and I must compliment you on the documentation. It is the **best** of its kind that I have seen."*

From an INSURANCE AGENCY MANAGER:

*"I would like to compliment you on... **an excellent... program.**"*

From a HOBBYIST USER out in Oklahoma:

*"I thought I would drop a quick note about QUIKPRO. I have it running and it **will do what I bought it for.** I am very pleased with it..."*

This from a Vice-President of a Federal Savings & Loan:

*"In the past several weeks, I have used QUIKPRO software on four **different** programs. I am pleased with the results achieved so far. With your help over the telephone, I have created an /ISI file for a **large** data file that was **already** in existence, and the data file now works with an input program created by QUIKPRO."*

## **Proven and Widely Used**

**Businesses, Schools, Hobbyists and Government are among our thousands of users...**

Johns Hopkins  
U.S. Department of  
Agriculture  
Proctor & Gamble  
Federal Express  
American Express  
Monsanto  
NASA  
Blue Cross Blue Shield  
Ford Motor Company  
Duracell International  
Westinghouse  
General Electric  
Random House  
U.S. Navy  
Tandy Corporation  
NCR  
DuPont  
RCA  
Satellite Broadcasting  
New York University

Brooklyn High School  
Blue Ridge School  
District  
Public Schools of  
Grand Rapids  
University of Alabama  
Exxon  
AT&T  
Texas Tech  
Clemson University  
U.S. Dept. of Energy  
U.S. EPA  
University of Maryland  
Mobil Chemical  
University of Arkansas  
University of Tennessee  
Speed Queen Co.  
Rhode Island Hospital  
University of Oklahoma  
University of Hartford  
Many, many more...

**Special Discount Offer**  
**ORDER NOW**

NOW! Your IBM, Tandy,  
TRS-80, Apple, Commodore,  
or Compatible is . . .

**1/2 OFF  
or more**

**a COMPUTER that  
WRITES PROGRAMS  
FOR YOU for 1/2 OFF\***

**ORDER NOW—Take Advantage of this  
SPECIAL OVER 1/2 OFF DISCOUNT OFFER**

Yes, send me QUIKPRO + II for my . . .

Check your computer type:

Item No.	Reg Price
<input type="checkbox"/> (1015) IBM PC, XT, AT	\$149
<input type="checkbox"/> (1015) IBM Compatible	149
<input type="checkbox"/> (1015) TANDY 1000, 1200	149
<input type="checkbox"/> (1015) TANDY 3000	149
<input type="checkbox"/> (1017) Apple IIc, IIe or Compatible	149
<input type="checkbox"/> (1016) Commodore 64 with Disk	149
<input type="checkbox"/> (1014) TRS-80 Model 4	149
<input type="checkbox"/> (1022) COLOR COMPUTER II or III (Radio Shack)	149
<input type="checkbox"/> (1013) TRS-80 Model 3	149
<input type="checkbox"/> (1011) TRS-80 Model 1	149
<input type="checkbox"/> (1012) TRS-80 Model 2, 12, or 16	149
<input type="checkbox"/> (1018) OSBORNE 1	149
<input type="checkbox"/> (1020) KAYPRO 2	149
<input type="checkbox"/> (1023) KAYPRO 4	149

**YOU  
PAY  
\$29.50**

for any  
computer  
type

Limited  
Offer  
Good  
For  
30  
Days

**SHIPPING & HANDLING  
TOTAL ORDER**

\$ 4.50  
\$ \_\_\_\_\_

YOU MAY ORDER BY MAIL OR

**CALL TOLL FREE 24 HOURS  
1-800-221-3333, Operator K885**

Payment By:  VISA/MasterCard  Check or Money Order  
 COD  Bill My Company (must be D&B rated & have Company P.O.)

VISA/MC # \_\_\_\_\_ Exp. Date \_\_\_\_\_

By Mail: Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Send To:

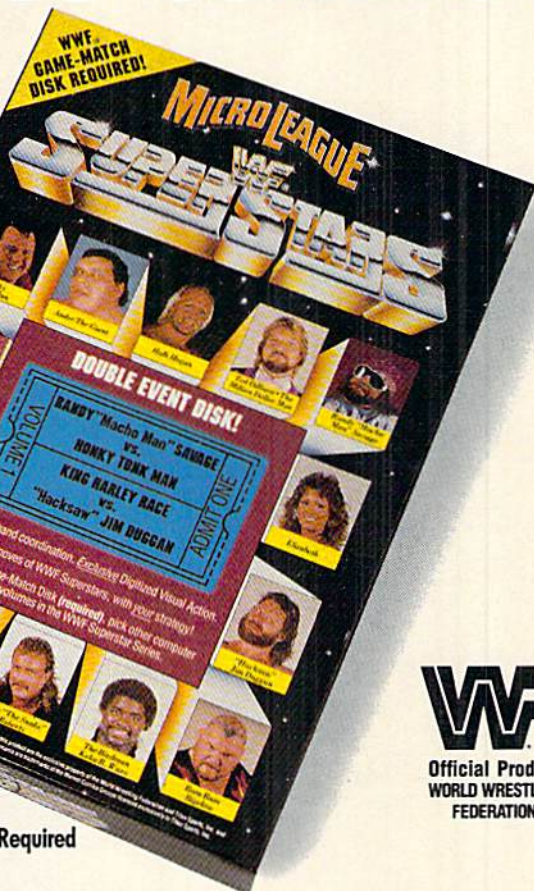
**ICR Future Soft**  
PO Box 1446-DI  
Orange Park, FL 32073

# NOW ENTERING THE RING... A WWF® WINNING COMBO!

Game-Match Disk  
Required



Commodore 64/128  
Color TV or Computer Monitor Required  
For 1-2 Players



"This time, McMahon, you got that right!" Coming from the independent-minded WWF® Superstar & announcer, Jesse "The Body" Ventura, that remark rings loud like the match bell.

On the left, the original Game-Match Disk featuring Hulk Hogan™ confronting WWF mainstays. Interviews where battle lines are drawn, realistic ringside commentary, and, of course, slam-bang *authentic* WWF footage in our unique DVA (Digitized Visual Action). On the right, introducing the newest entry—the WWF Superstar Series!

*Now*, the action and managerial challenges you'll face continue with other Main-Event matches. After powering up the Game-Match Disk (Required), you then choose from *two* titanic matches in *each* volume of the Superstar Series—

**VOL. 1** The Honky Tonk Man vs. "Macho Man" Randy Savage, who has vowed to protect the honor of the lovely Elizabeth. But with the talents he has, the Honky Tonk Man has lots to croon about! AND in a no-love-lost match, King Harley

Race tries to make sawdust of "Hacksaw" Jim Duggan!

**VOL. 2** The Hulkster™ tangles with the Million Dollar Man, Ted DiBiase, who won't be able to buy a title this time. He'll have to earn it the old-fashioned way—*inside* the

squared circle! PLUS experience the hot-blooded Ravishing Rick Rude entwined with Jake "The Snake" Roberts—whose cold-blooded reptilian friend can cool down the passions of many a foe.

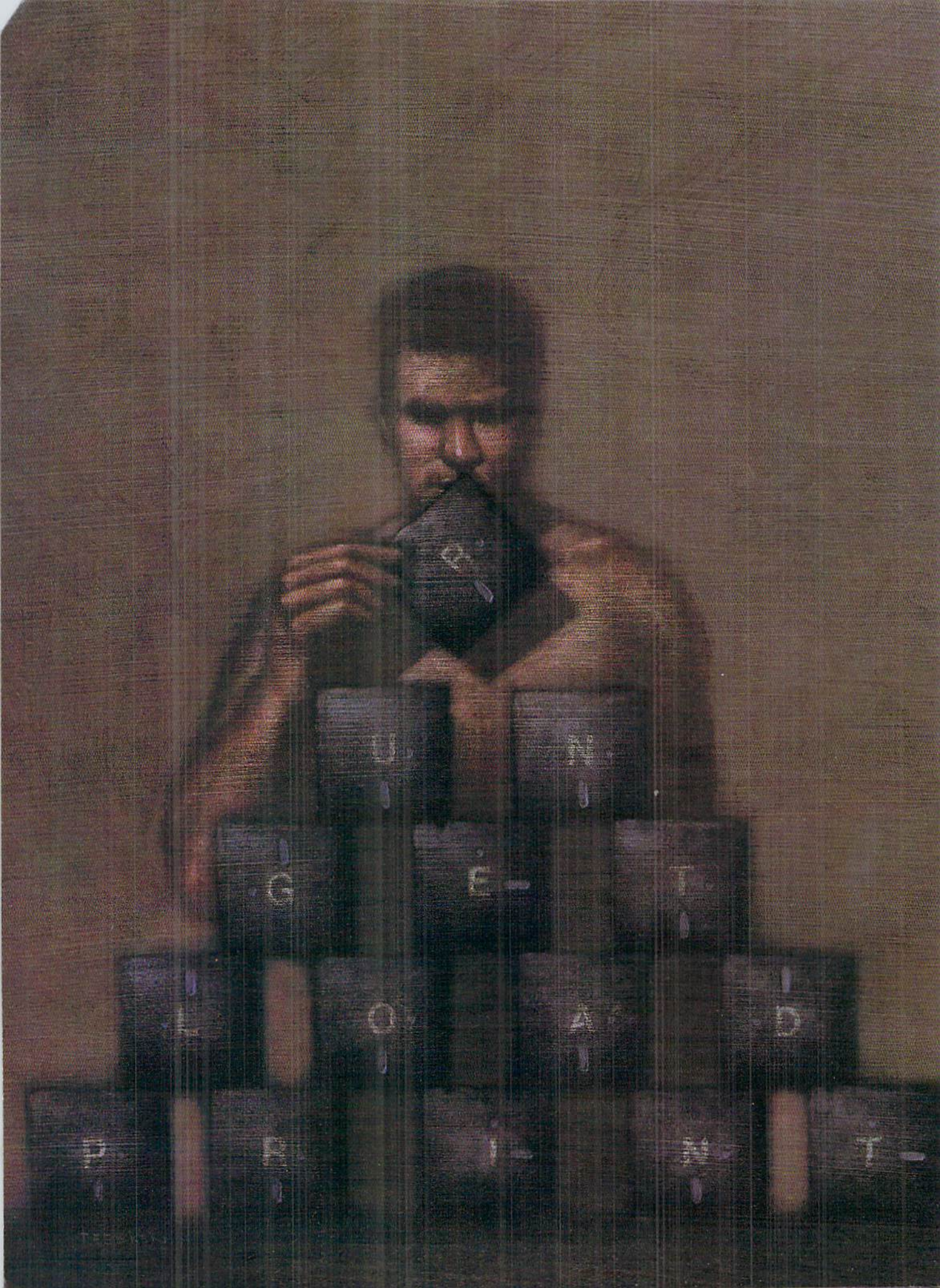
Micro League's WWF Wrestling. Not eye-hand coordination. Authentic, patented moves—where *your* managerial wizardry shapes the outcomes. With the Game-Match Disk (sugg. retail \$29.95), you'll enter the computer ring. With the Superstar Series (sugg. retail \$19.95), you'll be pinned to the screen!

If your retailer is out of WWF Disks, don't do the Shake, Rattle & Roll on the hapless clerk. Just call us at (302) 368-9990 for VISA/MC orders or drop us a (clothes) line to:

**MicroLeague Sports**  
2201 Drummond Plaza  
Newark, DE 19711



The likenesses, logos, copyrights, registered trademarks and WWF-related phrases depicted on this product are the exclusive property of the World Wrestling Federation and Titan Sports, Inc. and may not be reproduced without their express written consent. Hulk Hogan, Hulkster and Hulkamani are trademarks of the Marvel Comics Groups licensed exclusively to Titan Sports, Inc.



# Basic 101

*The best way to get to know your computer is to take the programming plunge.*



By ANNETTE HINSHAW

If you want to know your computer better, try a little programming. A few brushes with the dread "syntax error" will have a salutary effect on your understanding of how computers work, and that knowledge will transfer when you need to figure out why a program like a spreadsheet suddenly does something you don't expect.

This article introduces some programming fundamentals. The examples are in Basic, but the programming mechanisms are used in all computer languages. You can build on what you learn here with books and magazine articles devoted to teaching this important way to interface with a computer. If you belong to a users' group, that's another good place to look for help.

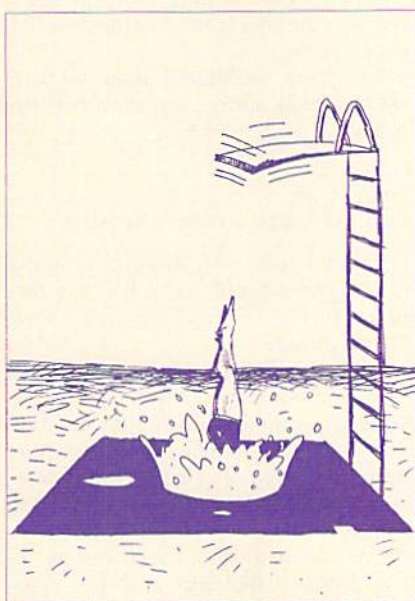
The only way to learn programming is to do it, so turn on your computer and, as you read, type in and try the examples. I don't have room for a detailed explanation of each computer command, so look up everything we cover in the user's manual that came with your machine. Try to relate what you read there to what you see happening on your screen.

In the examples, substitute your name every place you see "Annette." In the text of the article, <return> means to press the return key.

## WHAT IS A PROGRAM?

A program is a numbered list of instructions to the computer. Each instruction, also called a statement, is composed of computer commands in a specified order. You probably already use commands like Load, Run and Open.

When you load or run a program or open a channel to the disk drive, you're talking to your computer in Direct mode. As soon as you type the command and press return, the computer obeys the instruction and then forgets it. For example, type PRINT "ANNETTE" <re-



turn>. The computer will oblige, but to make it print the name again, you must retype the command.

In Program mode, statements are assembled into a program that the computer remembers, but it doesn't execute the commands until you run the program. Each statement in the program has a line number. Returning to our example, if you type 10 PRINT "ANNETTE" <return>, nothing happens unless you type RUN <return>. By giving the command a line number (10), you turned it into a program, which can be used over and over without retyping the line. To see your program, type LIST <return>.

## VARIABLES

Note that you have to type *exactly* what I say, or the program won't work correctly. For instance, if you leave out the quotation marks around the name, the computer will print a zero instead. Try it. How does ANNETTE become 0? Well, that takes a little explanation.

First of all, the computer deals with two kinds of data: numbers and strings. Numbers have specific meaning to the computer, which can manipulate them in computations. Strings, however, are merely sequences of characters (including numerals, letters or symbols) that the computer will faithfully reproduce on command, exactly as they were entered, without doing anything with them that you haven't specifically commanded. The computer recognizes as a string anything that is included within quotation marks, so you define a string by placing a sequence of characters in quotes.

For convenience in handling them within a program, both numbers and strings can be represented by letter symbols, which are called variables, because their value or content can change during program execution.

Number variable names can consist of one or two letters in any combination or any word not reserved by the computer language. To make a string variable name, add a dollar sign to the end of a number variable name—ANNETTE\$, for instance. Look up "variables" in your manual to find the rules for naming them.

Back to our question. Since ANNETTE is not in quotes and has no dollar sign after it, the computer assumes it's a number variable name, not a string or a string variable name. Finally, if the computer hasn't been told the value of a number variable, it assigns a value of zero.

You can define a number variable by assigning it a value with a statement like ANNETTE=10 <return>. Type that statement, followed by PRINT ANNETTE <return>, and the computer will return a ten.

In writing a program, it's important to distinguish carefully between number and string variable names. The computer gets upset if you ask it to do a ▶

## PROGRAMMING

computation with a string variable or manipulate a number variable like a string. (These mistakes give a Type Mismatch error.)

Let's try a little program using variables. As you type it in, remember that even punctuation is important, and you must press return after each line.

```
10 INPUT "TYPE A NUMBER"; ZZ
20 INPUT "TYPE ANOTHER NUMBER"
   ; YY
30 PRINT ZZ + YY
40 INPUT "TYPE A NAME"; A$
50 INPUT "TYPE ANOTHER NAME"; B$
60 PRINT A$ + " " + B$
```

Now, run the program to see how it works. The Input commands display a ? prompt on the screen and instruct the computer to wait until the user types something ending with a return. The message in quotes appears in front of the question mark to tell the user what information is needed, and the user's input is stored in the variable named after the semicolon. This input is one place where quotes aren't necessary to define a string; if the variable name indicates a string (if \$ follows the name), the computer stores the input as a string.

In lines 30 and 60, the computer uses the variables in Print statements. The user can type in different numbers and names each time the program is run, and the current values of the variables will always be the last ones typed. Notice the space between the quotation marks in line 60. It's needed to separate the strings in the output. Since a computer doesn't "understand" strings, you have to specify every character, even spaces.

You can change program lines or add more anytime you want. There are two ways to change a line. You can replace it by retyping it with the line number and pressing return, or you can edit it, with the help of the insert-delete key, and press return while the cursor is still on the corrected line. To view and edit any line that's not on the screen, type LIST <line number>.

If you run into a syntax error, check your typing. You have a misspelling, a punctuation mark missing or misplaced, or your commands are incomplete or in the wrong order. Remember, every character is significant.

You may also encounter a problem in what is called Quote mode; i.e., when the cursor is within a quote on a program line. In this situation, if you press a special key, such as a cursor key, the insert-delete key or a function key, the computer will print a graphics character instead of executing whatever the

key is designed to do. This means that you in effect erase whatever is overprinted by the graphics characters when you try, for example, to move the cursor to a character you want to change. If you get snarled like this on a line in Quote mode, hold down the shift key as you press return; a shifted return does not store the line, so you get out of the fix and can retype the line in corrected form.

To add a new line, type it with a so-far unused line number that places it where it belongs, and press return. The computer automatically arranges lines by number, no matter when you enter them, so the next time you list your program, the new line will appear in its proper place.

Type these additional lines for the little program above, and then list the program:

```
32 AB = ZZ - YY + 219
33 PRINT AB
34 C$ = "THIS IS A TEST":PRINT C$
```

The new lines will appear between lines 30 and 40, and, when you run the program, they'll produce two more lines of output.

Notice that in line 34 the colon separates two statements. When you're programming, sometimes you may need to put two or more statements on the same line to save memory or to control program logic at a branch. Use the colon to do this.

If you want to save this program to disk, choose a filename and then type SAVE "filename",8 <return> in Direct mode on the C-64 or DSAVE "filename" <return> in the same mode on the C-128. Look up rules for filenames.

After you've saved the program, type NEW <return>. This command erases the program from the computer's memory and resets the variables to zero. Always NEW the memory before you begin another program to avoid getting lines from the old program entangled in the new.

### LOOPS AND BRANCHES

Most programs are composed of tiny programs of one or more lines called subroutines, each of which accomplishes one specific operation. These operations, in turn, build and are built from a few fundamental programming mechanisms.

Two of the most important of these mechanisms are loops and branches, and they're closely related. Unless the computer is told otherwise, it executes program lines in numerical sequence, so if you want execution to go elsewhere

than the next line, you must set up a branch operation. Statements used for branching include GoTo and If-Then-Else. (Else is not available in Basic 2.0 on the C-64.)

A loop is a special kind of branch that turns on itself, repeating one or more times. Just how often it repeats depends on what Basic statements you use and how you combine them. I can't show you here all the ways to set up a loop, but statements such as For-Next, GoTo, GoSub and, on the C-128, Do-Loop are used.

The simplest loop is an endless loop. Try this one:

```
10 PRINT "ANNETTE ";
50 GOTO 10
```

Pretty heady, isn't it, seeing your name march across the screen? Each time the computer comes to line 50, it's told to go to line 10 and start executing, which, of course, takes it to line 50 again. The only way to stop the program is to press the run-stop key.

The semicolon is what makes the computer continue printing on the same line. Ordinarily, the computer executes a return for every Print statement, but the semicolon suppresses the return. A comma in the same place moves the cursor to a predetermined column position. Try it.

An endless loop is boring after a while, so let's put a limit on this program. Type the following lines:

```
20 X = X + 1
30 PRINT X,
40 IF X = 100 THEN END
```

Now, when you run the program, the computer will print your name 100 times. Line 20 defines X as a counter, whose value increases by one each time through the loop. In the If statement in line 40, the computer tests to see if X is 100 yet. If not, the loop continues. When the value of X finally becomes 100, the computer does whatever follows THEN. You could, for example, say THEN GOTO 100 and start a new subroutine at line 100.

A For-Next statement is another way to set up a loop in Basic. NEW the memory, then type:

```
10 N$ = "ANNETTE"
20 FOR C = 149 TO 155
30 PRINT CHR$(C);
40 PRINT N$;C
50 NEXT
60 PRINT C
```

The loop starts with the For in line 20. The initial value of C—the counter for the loop—is the first of the two



## PROGRAMMING

numbers following the equals sign. Unless you specify a different STEP (increment), the computer will add one to C's value each time execution reaches the Next statement in line 50—the end of the loop. As long as C remains less than the second number, execution will return from the Next statement to the For statement. In other words, the lines from FOR to NEXT are executed repeatedly until C = 156. Then the com-

puter goes on to the line after NEXT. Each loop in this example changes the color of the characters displayed on the screen.

### PUTTING IT ALL TOGETHER

Writing programs is a matter of putting operations such as loops and branches together in more or less complex patterns, and the most important consideration in programming is the

sequence of these operations. One of the best ways to plan a program is to make a list of what you want your program to do—that is, *define your output*. Then draw a flowchart, a diagram of program logic that shows each operation, including branches, in order of execution.

For instance, before writing the simple dice-throwing program (see Listing 1), I identified three goals:

—To be able to "throw the dice" any number of times without typing RUN every time.

—To have the computer throw two six-sided dice.

—To have the computer tell me if I "won" (threw a 7 or 11) or "lost."

Each of these features calls for a specific operation. The first requires a loop with a branch point, where the alternatives are to play again or to quit. The second calls for using the random number generator to simulate dice. The third needs a test of the dice throw and a branch leading to the appropriate Print statement.

Study the flowchart (Figure 1) and then the program itself. The flowchart makes the sequence of the program logic apparent, even though you may not know the commands necessary to actually write the program. By the way, the REM statements in the listing are notes for the programmer; the computer ignores them.

The best way to learn Basic programming is to study working programs. Find short ones in magazines or books or in the public domain; then analyze every line. Look up each command and try to see how it contributes to the program; draw a flowchart of the program's logic and play with any statements you don't understand; and feel free to edit, delete and re-arrange lines. You can also insert Stop statements here and there, or add Print statements to make the computer display what happens when you use a function like RND(X). In other words, play with the program until its changed behavior tells you how each statement works and fits into the overall logic. If the program stops working, figure out why. If the computer freezes, just turn it off and reload.

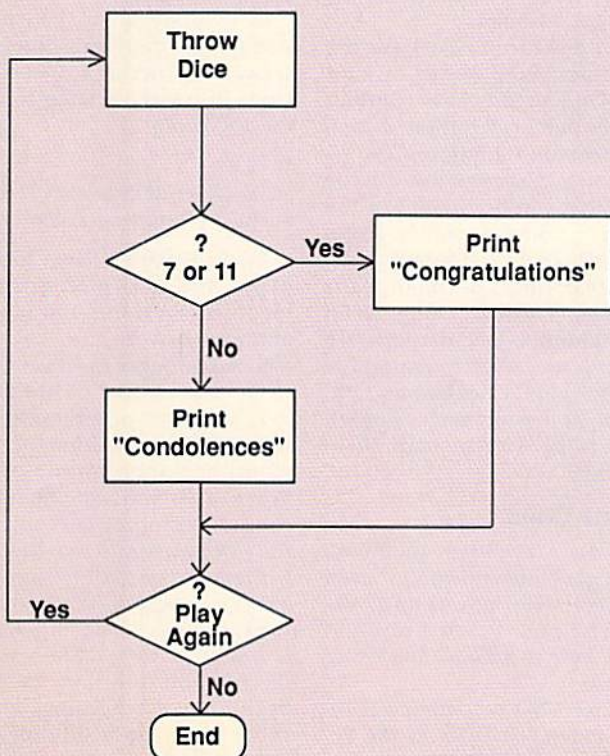
Keep this process up, and one day you'll either be a good programmer or you'll know that programming isn't your cup of tea. Whichever way it goes, you'll understand your computer a lot better, and you'll trust it a lot more. ■

Annette Hinshaw, founder of the Tulsa Area Commodore User's Group, has written extensively for computer magazines.

Listing 1. Dice game program.

```
5 REM DICE THROWING PROGRAM
10 REM CLEAR VARIABLES
20 CLR
25 REM THROW TWO SIX-SIDED DICE
30 DI=INT(RND(0)*6+1)
40 D2=INT(RND(0)*6+1)
50 PRINT "DIE 1=" D1;"DIE 2=" D2
55 REM BRANCH IF NOT 7 OR 11
60 IF D1+D2=7 OR D1+D2=11 THEN GOTO 90
70 PRINT "TOO BAD!"
80 GOTO 100
90 PRINT "YOU WON!"
100 INPUT "WANT TO PLAY AGAIN (Y/N)";I$
105 REM TEST ANSWER ON REPLAY
110 IF LEFT$(I$,1)<>"Y" THEN END
115 REM LOOP BACK TO DICE THROW
120 GOTO 20
```

Figure 1. Dice game flowchart.



# — The Secret — Of Better Programming

*Whether you're programming in Basic or machine language,  
these guidelines will help smooth the way.*



By JOHN RYAN

**W**riting a computer program, like writing a novel or magazine article, is an art, and, like a writer, each programmer develops his or her own style. However, all programmers can benefit from following basic guidelines, and, in this article, I'll present those I've found of help. If you follow them, too, you'll be well on your way toward the efficient production of interesting and smoothly running programs.

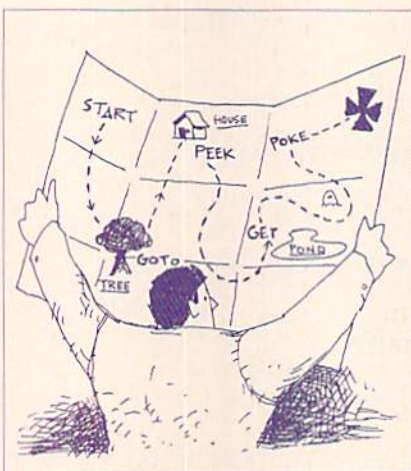
## PLANNING

Programming is often ten percent passionate coding and 90 percent very tedious debugging. Good planning, however, if followed through to its logical conclusion, can even out these percentages.

Planning involves creating some type of map to follow. This map may be anything from simple notes on the logic for a particular section of your program to a formal flowchart for the entire thing. I don't think most projects require textbook-style flowcharting, but some sort of thoughtful overall mapping is surely necessary to avoid chaos if your program is to be at all complex.

The following are the points that I usually sketch out before turning on my computer:

1. What is the finished program supposed to do? If it's a game, when does play end?
2. What are the main sections of the program? Score-keeping, sound routines, menus, joystick routines, mathematical calculations, text handling? Whatever they are, I write these down in the order they'll appear in the listing.
3. For games, what are the rules? I write down point values and assign variable names to them. If using a redefined character set, I assign the screen codes to be used.
4. I write down the memory addresses of all storage areas, whether they



be for sprites, graphics screens or machine language variables.

5. Finally, I make sure I can write the program without being slowed by a lot of research. Do I already have routines written earlier that I can import? If not, what references am I going to need?

The foregoing describes just a basic list of items that can be helpful in the heat of battle. It could be several pages longer (I've filled several spiral notebooks with planning information for some projects). However long, keep your programming notes in a separate notebook and use scratch paper for quick logic flows and calculations. Otherwise, you'll find your neat program information filled with incomprehensible doodlings.

## WRITING THE CODE

If you've done adequate planning, you'll have a good idea how to proceed with coding. However, here again guidelines can be helpful. Below I've listed the steps that I use to make code writing more effective.

1. No matter what the language, define all the major variables at the beginning of the listing. Also, ensure that Dimension (DIM) statements are out-

side the main program loop, to avoid redimension errors.

2. Number lines by increments of at least ten. By doing so, you can make additions and corrections to your program without using a renumbering utility. Only when the program is finished and completely debugged will you need to use a renumbering utility or command to number the lines evenly.

3. You should *never* start a target subroutine or line with a Remark statement. Put the REM above the target line so it can be deleted to save memory and increase speed.

4. Keep two copies of your finished program, one liberally annotated and one pared down, with all the REM statements removed.

5. In addition to leaving out REM statements, there are other ways to increase a program's speed. Stay away from indexed variable loops, such as the following:

```
10 X = 0
20 X = X + 1: IF X <> 1000 THEN 20
30 REM Continue execution here.
```

Instead, use For-Next loops as much as possible. Also, placing multiple statements on each line will save both time and memory.

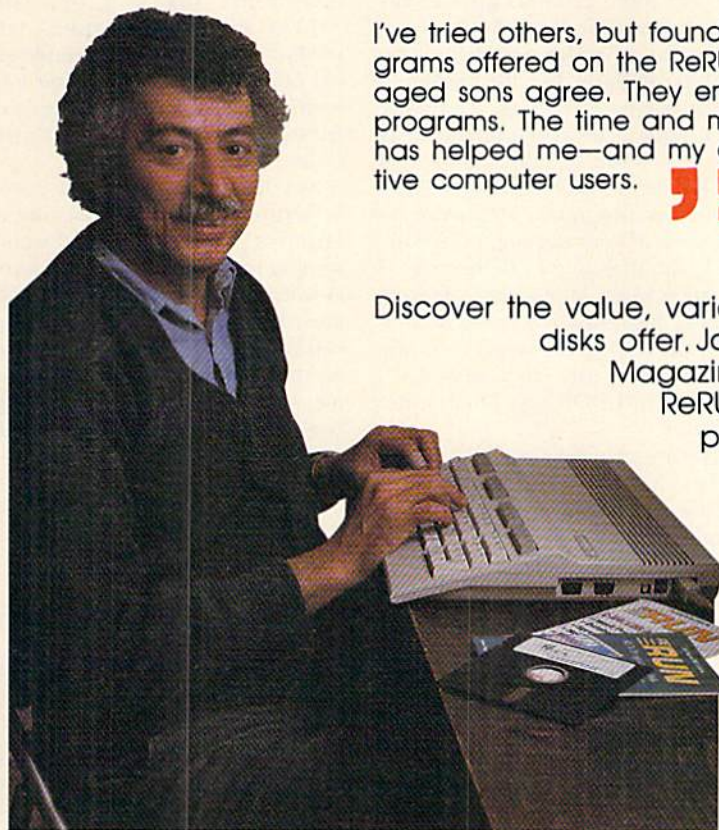
6. Use integer variables for constants (A% = 100 instead of A = 100).

7. For large programs, it may be better to place subroutines and Data statements at the beginning of the listing. When a GoSub statement calls a subroutine beginning later in the program, the computer must search forward until it finds the target line number, and in a large program, the search time can really add up. If the target line number is less than the GoSub's line number, the program will start searching at the beginning of the listing, and, if the target is near the beginning, the search will be short.

8. C-64 owners can minimize irritat- ▶

NOW ONLY  
\$69.97 A YEAR!

“ For my money, RERUN DISKS are the greatest.



I've tried others, but found that the quality and number of programs offered on the ReRUN disk can't be beat. My two teen-aged sons agree. They enjoy the games and educational programs. The time and money that I've saved with ReRUN has helped me—and my entire family—become more productive computer users.



—Jim Palmieri, ReRUN Subscriber  
Farmingville, NY

Discover the value, variety and ease-of-use that ReRUN disks offer. Join the thousands of RUN Magazine readers who subscribe to ReRUN. Each disk is packed with programs from the two most recent issues of RUN, plus never-before-published BONUS programs. ReRUN is great software at an affordable price, including:

- Word Processing
- Spreadsheets
- Data Bases
- Educational Applications
- Home Entertainment

ORDER A SUBSCRIPTION TODAY! CALL TOLL-FREE **1-800-343-0728**

(single issues available at \$16.47 each)



## SAVE 30% ON A YEAR'S SUBSCRIPTION

**YES!** I want to save time and money! Send me the following:

- One year (6 issue) subscription to ReRUN for only \$69.97  
 Nov./Dec. '88 single issue for \$16.47  
 Back issues at \$16.47 each

month \_\_\_\_\_ year \_\_\_\_\_  
month \_\_\_\_\_ year \_\_\_\_\_

- Payment Enclosed       VISA  
 MasterCard       American Express

Card # \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Add \$23.70 for foreign airmail  
Please allow up to 2 weeks for delivery  
mail to: RRRS8  
RERUN 80 ELM ST. PETERBOROUGH, NH 03458

# RE RUN

## PROGRAMMING

ing delays by adding a FRE(0) command at the beginning of string-handling routines to force garbage collection before variable space becomes cluttered. Garbage collection occurs when there's no room for a new variable and everything must be shifted from the top of memory down to make room.

9. Save your program frequently during development, and make backup copies on a separate disk that's set aside for safekeeping. If you own a printer, also print out a listing of the finished program for your archives; paper lasts longer than disks!

10. Each time you work on a program, place the current date, time and version number in a REM statement at the beginning of the listing. Frequently, I find myself with several versions of a program and would have a difficult time knowing which was the most recent if I hadn't taken this precaution.

11. After you've finished your program, keep some sort of log (paper or disk) of its major routines, so you can find them to use in other programs.

After a while, you'll be surprised at how many subroutines you'll have on tap.

### MACHINE LANGUAGE PROGRAMMING

Here are some pointers specifically for writing machine language code:

1. "Top-down" programming may be acceptable for Basic, but it's hardly efficient for machine language. Use modular techniques, constructing your program from a series of subroutines, and make sure they're global enough to be used in other programs. Moreover, by passing most of your parameters to subroutines, you'll find that debugging is easier, since you'll know what registers or memory locations are being manipulated by the calling and target routines.

2. Use meaningful label names. To me, JOYSTICK'LOOP says much more than JLOOP.

3. Likewise, annotate your listing liberally. It can be a headache trying to determine what a particular routine was designed for after you haven't seen it for several months. Unlike Basic, ma-

chine language comments don't appear in the assembled object code and don't affect execution speed, so programmers have no excuse for not using them.

4. Start a module library of handy subroutines that can be .LIBed or .FILEed into new programs. Also, get used to using standard variable and Kernal names in the declaration table and saving them as a Library file to disk. In addition, use macros if your assembler system can handle them.

Needless to say, an entire book could be written on how to become a more effective programmer, and what works for you may not work for someone else. However, I've found that following these simple guidelines has let me concentrate more on actual coding and less on the mechanics of programming. For me, that has meant the difference between programs that work well and programs that merely work. ■

*John Ryan is an air traffic control instructor and an advanced programmer in both Basic and machine language.*



## Excellence ... for the Commodore

**Lt. Kernal** - a 20 or 40 Megabyte Hard Drive which supports CPM, includes enhanced system commands, and is expandable, configurable, & FAST! Great for BBS operation.

**Super Graphix GOLD** - the ultimate printer interface which supports 128 FAST serial and includes a 32K buffer, 4 built-in fonts, 4 downloadable fonts, and a utility disk with 27 fonts.

**Super Graphix** - an enhanced printer interface which has NLQ built in and includes an 8K buffer, 2 downloadable fonts, reset button, and a utility disk with 27 fonts.

**Super Graphix jr** - an economical printer interface with NLQ built in and includes 10 printing modes, graphics, and easy operation.

**FontMaster II** - a powerful word processor for the C-64 with 30 fonts ready to use, 65 commands, font creator, data merging, super- and subscripting, italicizing and more.

**FontMaster 128** - a super word processor for the 128 with 56 fonts ready to use including foreign language fonts, on-screen font preview, 4 column printing, a 102,000-word spell checker and much more.

The Xetec Product Family for the Commodore C64® and 128® .

The name that spells *Quality, Affordability, and Reliability*

All Hardware is FCC Certified    All Interfaces include a Lifetime Warranty

**xetec**

Commodore C64 and 128 are registered trademarks of Commodore Business Machines, Inc.  
Xetec, Inc. 2804 Arnold Rd. Salina, KS. 67401 (913) 827-0685

# C-64 Sprite Basic

*What was once tedium on the C-64 becomes a joy with the powerful new sprite commands in this language extension.*



By CHARLES ORCUTT

**A**s one of the outstanding aspects of graphics on the C-64 and C-128, sprites are used in programs in a variety of ways, from arcade games to the pointer found in "point and click" interfaces like GEOS. Since both computers have essentially the same VIC-II graphics chip, they have equal abilities when it comes to sprites.

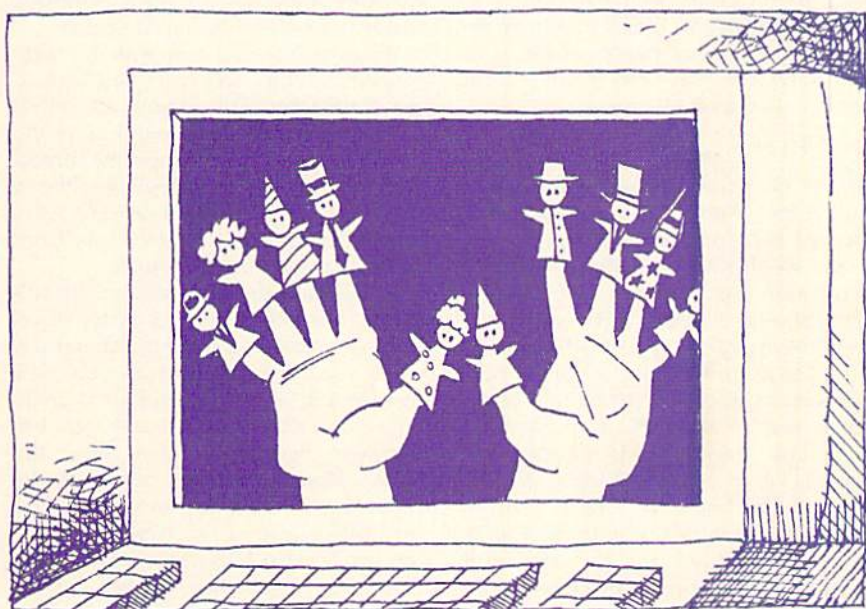
But there are differences between them when it comes to programming sprites. It is actually a little easier to program sprites from machine language on the C-64 than on the C-128, since you don't have to worry about things like shadow registers or multiple banks of RAM.

From Basic, however, sprite programming is far easier on the 128 than on the 64, since Basic 7.0 on the C-128 has many dedicated sprite commands, and the C-64's Basic 2.0 has none. C-64 Basic programmers must therefore resort to Peeks and Pokes, which are slow, cryptic and in most cases too complex for the average programmer. Of course, there are commercial Basic extensions that add sprite commands (Simons' Basic, Super Expander, Graphics Basic), but these cost money and limit the use of the programs to those who possess the extension that was employed.

## SOLUTION EXTENDED

My solution was to write C-64 Sprite Basic, a small language extension that offers Basic programmers the most important abilities of the C-128's Basic 7.0 commands. Its great advantage is that a program that uses the language need only be copied to a disk, and anyone can run it.

There are four commands in the set. With them, you can define, move and even animate sprites with simple Basic



commands. One of them allows your program to automatically branch to a subroutine whenever a collision occurs between sprites or between sprites and screen data.

Here are the new commands Sprite Basic makes available. The first is

`SPRITE S#(1-8),On/Off(0/1),Color(1-16),  
Priority(0/1),XP(0/1),YP(0/1),Mode(0/1)`

The Sprite command is used to define a sprite. The sprite number (S#) is from 1-8, not the 0-7 you may have used when poking in sprite information from Basic 2.0. The second parameter turns the sprite on (1) or off (0). The sprite's color is next, with values from 1 (black) to 16 (light gray).

Then comes the priority flag, which determines whether the sprite will appear in front of or behind any non-sprite data on the screen. You can also expand the sprite horizontally or ver-

tically by placing a 1 in the appropriate XP or YP location. And, finally, you must indicate if the sprite is a normal, high-resolution, single-color sprite (Mode = 0), or a multicolor (Mode = 1).

All the various parameters in the Sprite command can differ for each of the eight sprites, so some can be high-resolution while others can be multicolor. Some can appear in front of screen data, others behind. The colors can be different, as can the X or Y expansion flags. It all depends on what you need in your program. As you can see, the Sprite command quickly and easily replaces the many different Pokes that would have been necessary to do the same things.

The second command is

`SPRCOLOR MC1 (1-16),MC2 (1-16)`

If any of the sprites in your program are multicolor, you need to use SPR-▶

# PROGRAMMING

COLOR to indicate which colors to use. Keep in mind that these colors are used by *all* your multicolor sprites. The only color unique to a single sprite is the one specified in the Sprite command itself.

The third command is

```
MOVSPR S#(1-8),X1,Y1[,X2,Y2,SPEED(1-8)]
```

MOVSPR is the real workhorse of the command set. If you've ever tried moving sprites by poking values into memory, you know how difficult it can be when you attempt to move past location 255. It requires a lot of complex peeking, poking, ANDing and ORing of memory locations. Well, you can forget all that, because MOVSPR handles it for you. There are two modes of use. If all you want is to place the sprite at some specific location, just use

```
MOVSPR S#,X,Y
```

where S# is the number of the sprite, and X and Y are the coordinates of its desired location on the screen.

But MOVSPR has another, more powerful animation feature. You can use the optional X2,Y2,SPEED parameters to move the sprite precisely from point X1,Y1 to point X2,Y2 at any of the eight allowable speeds. The thing that makes this aspect of MOVSPR so powerful is that while your sprite is moving, *Basic continues to process other parts of your program at the same time!* This is possible because this feature, which I call a sprite line, is executed on the interrupt, sixty times a second. One result is that you can automatically animate *all eight* sprites simultaneously!

Finally, there is the remarkable Collision command:

```
COLLISION TYPE (1/2),LINE #
```

When you are programming sprites, it is often useful to detect when one sprite has collided with another, or with some non-sprite data on the screen. With Basic 2.0, you would have to constantly peek memory locations, make a decision on what to do, and then branch your program to the line appropriate for handling it—altogether a complex, tedious and slow process. With the Collision command, however, you can tell

the computer to monitor for collisions, and if it detects one, to go to a specified subroutine. It will execute the subroutine, then return to where it left off and continue on.

To use the Collision command, you should have it on a line by itself somewhere in your program, preferably in the main loop. (If you're using both types of collisions, they should both be on the same line.) In your subroutine that handles the collision, the *very first statement* must be a Collision command (i.e., COLLISION,1 or COLLISION,2), but must *not* be followed by any line number. This will turn off collision detection while you are in the subroutine, and it is an absolute requirement!

When you're finished with the subroutine, a return will carry you back to the main loop. The reason the initial Collision statement(s) should be in the main loop is so that they can be turned back on after the return from a collision subroutine. Note: Be very sure you *never* turn on sprite collision detection from within a collision subroutine!

You should understand that the collisions are handled via a raster interrupt, 60 times a second. Therefore, when you turn off collision detection in your subroutine, automatic branching ceases, although the computer continues to detect collisions. Since the Sprite Basic extension automatically clears the collision registers when you turn off collision branching, the registers are cleared to zero for a sixtieth of a second. If you then intend to peek the collision registers to determine which sprites have collided, you should first do some small amount of Basic in order to give the computer time to catch the next collision. Updating variables or a very small For-Next loop (1 to 10) is sufficient.

## TIPS FOR USE

To use Sprite Basic, first type in Listing 1 (using RUN's Checksum program). This is a Basic listing in the form of hexadecimal Data statements. Before running it, save it with some simple name, like Listing 1. Your disk drive will run for a time while Listing 1 generates the actual Sprite Basic program. This

newly created binary file on your disk is the Sprite Basic extension.


To create programs with it, you first need to load and activate the Sprite Basic extension. This can be done with the following small boot program.

```
10 IF A=0 THEN A=1:LOAD"SPRITE
   BASIC",8,1
20 SYS 49152
30 NEW
```

After running this, you can write the new commands, and the C-64 will understand what they are. In fact, it's a good idea to have lines 10 and 20 as the first two lines in all your Sprite Basic programs. (Some of the new commands won't work as intended unless the SYS 49152 is executed *within* the program.)

To help you get started, I've included a demo (Listing 2). It uses all the Sprite commands to generate a simple but entertaining arcade game. To use it, first type in Listing 1 and generate Sprite Basic as instructed above. Then load and activate the Sprite Basic extension and enter Listing 2. By studying this game, you'll find useful examples of all the Sprite Basic commands.

While Sprite Basic does just about everything you need, there are some things you still have to do the "old" way. For example, you still have to create your sprites with a sprite editor, and the sprite data will have to be poked into memory. You also need to poke the sprite pointer into the appropriate sprite register. (Sprite pointers indicate where in memory the sprite data has been poked. These pointers to the data are poked into locations 2040-2047 for sprites 0-7, respectively.)

Finally, you need to peek the collision registers to see what sprite has collided, if you need that information. There are two sprite collision registers. For sprite-sprite collisions, peek location 53278; for sprite-data collisions, peek location 53279. For further information, you should refer to the *C-64 Programmers Reference Guide*. 

*Charles Orcutt is an electronics technician and a self-taught Basic and machine language programmer.*

## Listing 1. Language Generator program.

```
10 REM C64 HEX LOADER      :REM*24      40 IF LEN(A$)<62 THEN 120:REM*4      TS$(C$,1):L$=RIGHT$(C$,1)
20 OPEN 8,8,8,"SPRITE BASIC,P,W      50 B$=MID$(A$,1,20)+MID$(A$,22,      :REM*137
   "                                20)+MID$(A$,43,20)      :REM*208
30 READ A$:IF A$="-1" THEN CLOS      60 FOR I=1 TO 30      :REM*214
   E8:END                          :REM*186      70 C$=MID$(B$, (I*2)-1,2):H$=LEF
                                     T$(C$,1):L$=RIGHT$(C$,1)
                                     :REM*137
80 H=VAL(H$):IF H$>"9" THEN H=A
   SC(H$)-55      :REM*56
90 L=VAL(L$):IF L$>"9" THEN L=A
```

# PROGRAMMING

```

SC(L$)-55 :REM*73      08AAD20F7B760 :REM*108      DF8C685A7BDF9 :REM*55
100 BY=H*16+L:PRINT#8,CHR$(BY); 460 DATA 6838E93049FF2DE0C68D E 690 DATA C685A8BDFAC685A620B1 C
      :REM*138      0C660209EB78A38C901 9007C90
110 NEXT:GOTO 300 :REM*191      9B00334CE2C1A2 :REM*126      538BDF9C6DDFCC6D029 BDF8C6D
120 IF LEN(A$)<21 THEN B$=A$:GO 470 DATA 0E4C37A48DE8C6A901CA F 700 DATA C6DDFDC6D019BDF6C649 F
      TO 150 :REM*72      0090A4CE7C1A20B4C37 A48DE7C      F2DF3C68DF3C6ACEBC6 ADECC6D
130 IF LEN(A$)<42 THEN B$=LEFT$ 6207900C92CD0 :REM*37      0034CA7C64C45 :REM*255
      (A$,20)+RIGHT$(A$, (LEN(A$)-      480 DATA F1207300208AAD20F7B7 A 710 DATA C42076AC6902CDBD06C718 7
      21)):GOTO 150 :REM*230      D12D0C909D0F9ADE8C6 38E9010      D00C79D06C7BD07C769 009D07C
140 B$=LEFT$(A$,20)+MID$(A$,22, 490 DATA 8DEDC69900D0ADE7C649 F 720 DATA C79DF8C6BDF9C67D02C7 9
      20)+RIGHT$(A$,LEN(A$)-42)      F2D10D08D10D0A5158D EEC6F00      DF9C62076AC6F002B01D BD04C71
      :REM*113      9ADE7C60D10D0 :REM*143      87DFEC69D04C7 :REM*143
150 FOR I=1 TO LEN(B$)/2:REM*10 500 DATA 8D10D0207900C92CD011 2 730 DATA BD05077DFFC69D05C7BD F
160 C$=MID$(B$, (I*2)-1,2):H$=LE 07300209EB7ACE9C6C8 8A8DEFC      AC6187D03C79DFAC6CE F4C6D00
      FT$(C$,1):L$=RIGHT$(C$,1)      69900D0207900 :REM*218      EACEBC6ADECC6 :REM*165
      :REM*83      510 DATA C92CD07320CDC2AD12D0 C 740 DATA D0034CA7C64C45C44C6B C
170 H=VAL(H$):IF H$>"9" THEN H= 909D0F920730020EBB7 A5148DF      4C902F0102050C5ADDC C6857AA
      ASC(H$)-55 :REM*222      0C6A5158DF1C6 :REM*30      DDDC6857B4C34 :REM*32
180 L=VAL(L$):IF L$>"9" THEN L= 520 DATA 8EF2C6ACE8C688B9ADC6 A 750 DATA C52050C5ADDEC6857AAD D
      ASC(L$)-55 :REM*243      A989DF5C6ADE7C69DF6 C620DDC      FC6857BA9008DEAC6AD E1C6853
190 BY=H*16+L:PRINT#8,CHR$(BY); 5A0E7C60DF3C6 :REM*63      9ADE2C6853A20 :REM*253
      :REM*49      530 DATA 8DF3C6207900C92CD01E 2 760 DATA 83A82076BC5AD1ED0AD1F D
200 NEXT:GOTO 300 :REM*16 07300209EB78A38C901 9020C90      060AD1ED0AD1FD0A539 8DE3C6A
300 REM C64 SPRITE EXTENSION 9001C48ACE8C6 :REM*117      5A8DE4C6A57A :REM*36
      :REM*72      540 DATA 88B9ADC6AA689DF7C660 A 770 DATA 8DE5C6A57B8DE6C660AD E
310 DATA 00C0A900A00099F5C6C8 D 0FA8DF3C6A9348D0403 A9C08D0      CE8C688B9ADC6AAA001 9DF7C66      3C68539ADE4C6853AAD E5C6857
      503A9FA8D00603 :REM*67      0A20E4C37A4AD :REM*106      AADE6C6857B60 :REM*251
320 DATA A9C08D0703A92F8D0803 A 9C18D0903A9008DE0C6 2080C56 550 DATA E7C649FF2DF3C68DF3C6 6 780 DATA 78A90A8D12D0AD11D029 7
      0209FC5A67AA0 :REM*213      0209EB78A38E9013034 38C910B      F8D11D0A9818D1AD0A9 FF8D140
      0209FC5A67AA0 :REM*213      02F8D25D02079 :REM*143      3A9C38D150358 :REM*150
330 DATA 04840FBD0021007C9FF F 560 DATA 00C92CD014207300209E B 790 DATA 60A9808D1AD078A9318D 1
      03EE8D0F4C920F03785 08C922F      78A38E901301938C910 B0148D2      403A9EA8D15035860A5 A8F00CB
      055240F702DC9 :REM*39      6D060209EB78A :REM*219      DF6C60D10D08D      0F6C60D10D08D
340 DATA 3FD004A999D025C93090 0 570 DATA 38C9019007C909B0034C 1 800 DATA 10D04CC5BDF6C649FF 2
      4C93C901D8471A000084 0B88867      BC3A20E4C37A48DE8C6 A901CAF      D10D08D10D0BDF5C60A A8A5A79
      ACAC8E8BD0002 :REM*110      0040A4C20C38D :REM*189      900D0A5A6C899 :REM*115
350 DATA 38F99EA0F0F5C980D02F 0 580 DATA E7C649FF2D15D08D15D0 2 810 DATA 00D060ADEFC69DFAC6AD E
      50BA471E8C899F01C9 00F0383      07900C92CD013207300 209EB7E      DC69DF8C6ADECC69DF9 C6ADF2C
      8E93AF004C949 :REM*215      001D009AD15D0 :REM*249      69DFDC6ADF0C6 :REM*134
360 DATA D002850F38E955D0A085 0 590 DATA 0DE7C68D15D0207900C9 2 820 DATA 9DFBC6ADF1C69DFCC638 B
      8BD0020F0E0C508F0DC C899FB0      CD017207300209EB78A 38E9013      DFBC6FDF8C69D08C7BD FCC6FDF
      1E8D0F0A67AE6 :REM*201      70738C910B0B2 :REM*183      9C69D09C72079 :REM*180
370 DATA 0BC8B99DA010FAB99EA0 D 600 DATA ACE8C69926D0207900C9 2 830 DATA C69D01C7C901D02A900 9
      0B5F00FBD000210BD99 FD01C67      CD01EADE7C649FF2D1B D08D1BD      D02C7208CC6BD08C79D FEC6BD0
      BA9FF857A60A0 :REM*89      02077300209EB7 :REM*162      9C79DFFC638BD :REM*156
380 DATA 00B9BEC6D002C8E8BD00 0 610 DATA E001D009AD1BD00DE7C6 8 840 DATA FDC6FDFAC69D08C7A900 E
      238F9BEC6F0F5C980D0 04050BD      D1BD0207900C92CD01E ADE7C64      9009D09C72079C69D03 C7208C
      099A67AE60BC8 :REM*200      9FF2D1DD08D1D :REM*220      6BD08C79D00C7 :REM*100
390 DATA B9BDC610FAB9BEC6D0E0 F 620 DATA D0207300209EB7E001D0 0 850 DATA BDFEC69D04C7BDFFC69D 0
      0C6100F240F30BC9FF F007C9C      9AD1DD00DE7C68D1DD0 207900C      5C7BD00C79D06C7A900 9D07C76
      CB0064C24A74C :REM*255      92CD01EADE7C6 :REM*99      0BD05C7DD07C7 :REM*77
400 DATA F3A638E9CBA8449A0FF C 630 DATA 49FF2D17D08D17D02073 0 860 DATA D006BD04C7DD06C760BD 0
      AF008C8B9BEC610FA30 F5C8B9B      0209EB7E001D009AD17 D00DE7C      9C7300BF003A90160BD 08C7D0F
      EC630052047AB :REM*69      68D17D0207900 :REM*30      860A9FF60BD09 :REM*225
410 DATA D0F54CEFA6207900D00E A 640 DATA C92CD01EADE7C649FF2D 1 870 DATA C7101549FF9D09C7BD08 C
      DEAC6F009AE1ED0AE1F D02013C      CD08D1CD0207300209E B7E001D      749FF9D08C7FE08C7D0 03FE09C
      5207300204BC1 :REM*14      009AD1CD00DE7 :REM*143      76068A868AA68 :REM*63
420 DATA 4CAEA7C9CC9004C9D090 0 650 DATA C68D1CD060AD19D08D19 D 880 DATA 4000152A3F54697E93A8 6
      62079004CEDA738E9CC 0AAAABDB      02901D0034C31EAADE0 C6F0222      8C1CCC1D8C205C3434F 4C4C495
      7C648BDB6C648 :REM*217      901F00DAD1ED0 :REM*75      3494FCE4D4F56 :REM*48
430 DATA 4C7300209EB7E001F00C E 660 DATA F008A9018DEAC64C33C4 A 890 DATA 5350D2535052434F4C4F D
      002F008209FC5A20E4C 37A48A4      DE0C62902F00AAD1FD0 F005A90      25350524954C5000000 0000000
      8207900C92CD0 :REM*198      28DEAC6ADF3C6 :REM*108      0000000000000000 :REM*239
440 DATA 3B68480DE0C68DE0C668 C 670 DATA D0034CA7C6A0078CEBC6 A 900 DATA 00000000000000000000 0
      901D010207300A57A8D DCC6A57      9808DECC68CEBC6ADEC C6F0EB2      00000000000000000000 0000000
      B8DDDC64CAF1 :REM*36      DF3C6D0094EEC :REM*53      0000000000000000 :REM*1
450 DATA 207300A57A8DDECC6A57B 8 680 DATA C68810ED4CA7C64EECC6 C 910 DATA 00000000 :REM*93
      DDFC6A5398DE1C6A53A 8DE2C62      EEBC6B9ADC6AABDF7C6 8DF4C6B      920 DATA -1 :REM*13
    
```

# PROGRAMMING

**Listing 2. Sprite demo program.**

```

10 IF A=0 THEN A=1:LOAD "SPRITE BASIC",8,1
20 SYS49152:REM INITIALIZE EXTENSION
30 FORX=16128TO16383:READA:CK=C K+A:POKEX,A:NEXT
40 IFCK<>9111THENPRINT"ERROR IN DATA STATEMENTS":END
50 SID=54272:J=56320:V=53248
60 GOSUB750
70 POKESID+5,160:POKESID+6,252
80 POKESID+24,15
90 POKESID+12,103:POKESID+13,204
100 POKESID+8,40
110 POKESID+19,0:POKESID+20,253

120 POKESID+15,60
130 POKE53280,0:POKE53281,0
140 PRINT"(SHFT CLR)"
150 C1=5:C2=9:TL=1
160 POKE2040,252:REM CANNON
170 POKE2041,255:REM PLANE 1
180 POKE2042,253:REM MISSILE
190 SPRITE 1,1,6
200 SPRITE 2,1,13,0,0,0,1
210 SPRITE 3,0,7,0,0,0,1
220 MOVSPR 1,160,218
230 FORX=19TO0 STEP-1
240 POKE1024+X,160
250 POKE1063-X,160
260 POKE55296+X,2
270 POKE55335-X,2
280 POKE56256+X,3
290 POKE56295-X,3
300 POKE1984+X,160
310 POKE2023-X,160
320 NEXT
330 X=PEEK(V+30):X=PEEK(V+31):REM BE SURE ALL COLLIDES ARE CLEARED
340 R=INT(RND(1)*120):P=P+1
350 IF P=16THEN660
360 R2=INT(RND(1)*120)
370 S1=INT(RND(1)*7)+2
380 POKESID+1,S1*3
390 SPRITE 2,1
400 IFTL=1THENTL=2:POKE2041,254:MOVSPR 2,0,60+R,345,60+R2,S1:GOTO420
410 IFTL=2THENTL=1:POKE2041,255:MOVSPR 2,345,60+R,0,60+R2,S1
420 POKESID+4,129
430 C=C+1:IFC=30THENC=0:GOTO340

440 IFC=15THENPOKESID+4,128
450 C1=C1+1:IFC1=17THENC1=1
460 C2=C2+1:IFC2=17THENC2=1
470 SPRCOLOR C1,C2
480 PRINT"(HOME){CTRL 3}{CTRL 9}{6 SPACES}PASSES"P" MISSED"S" HITS"H
490 COLLISION 2,720:COLLISION 1,730
500 IFPEEK(J)AND8THEN530
510 IFFT=1THEN560
520 FT=1:GOSUB620:MOVSPR 1,X,218,320,218,2
530 IFPEEK(J)AND4THEN560
540 IFFT=2THEN560
550 FT=2:GOSUB620:MOVSPR 1,X,218,25,218,2
560 IFPEEK(J)AND16THEN430
570 IFF=1THEN430
580 IFF=0THEN F=1:GOSUB620:MOVSPR 3,X,193,X,55,3:SPRITE 3,1:POKESID+11,129
590 GOTO430:REM CONTINUE MAIN LOOP
600 REM THIS SUBROUTINE MAKES A VARIABLE EQUAL TO A SPRITE POSITION
610 REM IF YOU MEAN TO DO SPRITE 3 THEN IT IS X=PEEK(V+4)+PEEK((V+16)AND4)*64
620 POKE828,120:POKE829,96:SYS828:REM INTERRUPTS OFF
630 X=PEEK(V)+(PEEK(V+16)AND1)*256
640 POKE828,88:SYS828:REM INTERRUPTS ON
650 RETURN
660 POKE53280,7:POKE53281,9:PRINT"(SHFT CLR)":FORX=1TO3:SPRITE X,0:NEXT:GOSUB750
670 FORX=0TO12:PRINT:NEXT
680 PRINT"{CTRL 8}YOU HIT"H"WIT H"S+H"SHOTS IN THE 15":PRINT"PASSES OF THE ENEMY."
690 PRINT"PRESS FIRE FOR MORE OF THE SAME."
700 IFPEEK(J)AND16THEN700
710 CLR:GOTO50
720 COLLISION 2:SPRITE 3,0:F=0:MOVSPR3,X,193:S=S+1:POKESID+11,128:RETURN
730 COLLISION 1:H=H+1:SPRITE 2,0:SPRITE 3,0:MOVSPR 3,X,193:F=0:POKESID+1,0
740 POKESID+11,128:POKESID+18,129:POKESID+18,128:RETURN
750 FORL=0TO24:POKESID+L,0:NEXT:RETURN
760 REM CANNON DATA
770 DATA000,062,000,000,065,000,000,062
780 DATA000,000,034,000,000,034,000,000
790 DATA034,000,000,034,000,000,034,000
800 DATA001,255,192,001,255,192,001,255
810 DATA192,000,162,128,000,162,128,000
820 DATA162,128,000,162,128,000,162,128
830 DATA000,162,128,000,193,128,000,128
840 DATA128,007,000,112,007,255,240,000
850 REM MISSILE DATA
860 DATA000,008,000,008,000,000,000,042
870 DATA000,000,042,000,000,042,000,000
880 DATA042,000,000,042,000,000,042,000
890 DATA000,042,000,000,042,000,000,042
900 DATA000,000,128,128,002,132,160,002
910 DATA140,160,002,132,160,002,140,160
920 DATA000,055,000,000,012,000,000,000
930 DATA000,000,000,000,000,000,000,000
940 REM PLANE 1 DATA
950 DATA000,000,000,000,000,000,000,000
960 DATA000,000,000,000,000,000,000,000
970 DATA000,000,000,002,000,000,010,128
980 DATA008,040,032,222,170,170,010,138
990 DATA128,000,002,000,000,000,000,000
1000 DATA000,000,000,000,000,000,000,000
1010 DATA000,000,000,000,000,000,000,000
1020 DATA000,000,000,000,000,000,000,116
1030 DATA000,000,000,000,000,000,000,000
1040 REM PLANE 2 DATA
1050 DATA000,000,000,000,000,000,000,000
1060 DATA000,000,000,000,128,000,002,160,032
1070 DATA008,040,032,170,170,183,002,162
1080 DATA160,000,128,000,000,000,000,000
1090 DATA000,000,000,000,000,000,000,000
1100 DATA000,000,000,000,000,000,000,000
1110 DATA000,000,000,000,000,000,000,000
1120 DATA000,000,000,000,000,000,000,000
1130 DATA000,000,000,000,000,000,000,000
1140 DATA000,000,000,128,000,002,160,032
1150 DATA008,040,032,170,170,183,002,162
1160 DATA160,000,128,000,000,000,000,000
1170 DATA000,000,000,000,000,000,000,000
1180 DATA000,000,000,000,000,000,000,000
1190 DATA000,000,000,000,000,000,000,000

```



# C-128 Sprite Action

*Master the use of sprites with these high-level  
Basic 7.0 commands.*



By ROB KENNEDY

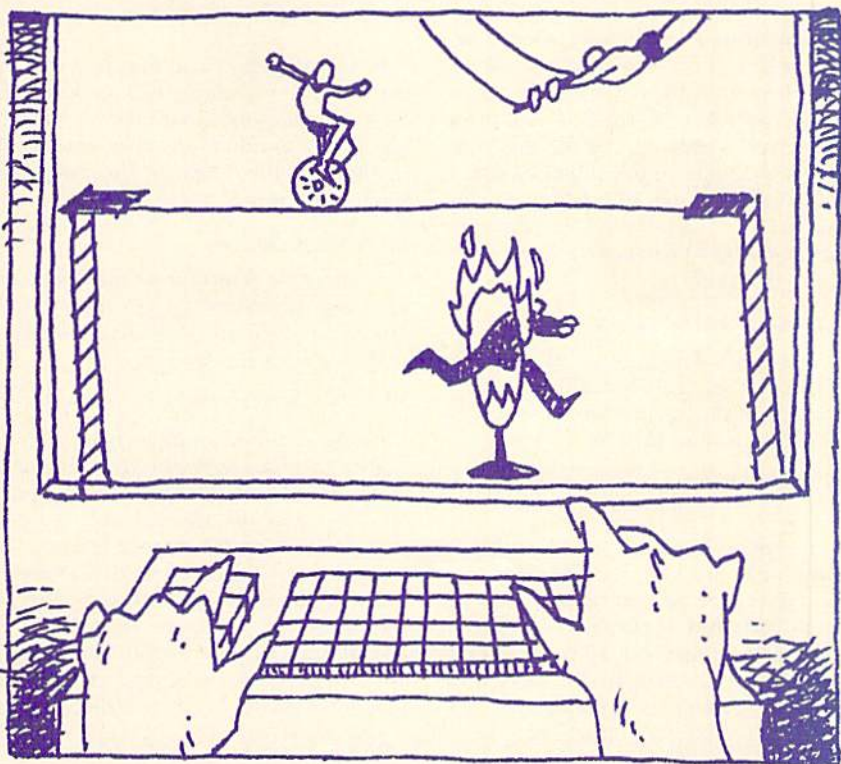
**B**asic 7.0 is the most advanced Basic language in any Commodore 8-bit computer for many reasons, one of which is its advanced sprite commands. On the C-64, sprites take hours of designing with pencil and graph paper, consulting reference books and typing in Poke commands, but with the C-128, you can create detailed multicolored sprites within minutes, and animation in just a little longer.

## THE SPRITE EDITOR

When I state that sprites can be created in minutes on the C-128, I'm not joking. Basic 7.0 has a command, SprDef, that activates a program for that specific purpose. Turn on your computer in 128 mode and 40 columns; then type SPRDEF and press return. Immediately, a large box will appear on the screen, along with a prompt asking which sprite you want to edit. Next, Type 1, and you'll see a 24-column x 21-line box containing a grid pattern, with, to its right, the sprite version of the same pattern. To clear the grid in both locations, press shift/clear-home.

Now find the little plus sign in the top-left corner of the box and try moving it around with the cursor keys. Other keys that control cursor position are the return key, which brings it to the beginning of the next line, and the home key, which brings it to the top-left corner of the box. Number keys 1-4 are used for drawing in the box. The 1 key erases, 2 draws in the standard color mode, and 3 and 4 draw in Multicolor mode.

Right now you're in the Standard mode, so you can draw with only one color. To change this color to any of the 16 colors available on the C-128, use the control and Commodore keys in conjunction with number keys 1-8.



Try drawing a shape in the box—just a simple one for now, like a rectangle. Notice, as you do so, that the sprite on the right is updated continually.

When drawing vertical lines, you'll probably find it cumbersome to hit a color key, then have to move down and left to get in position for pressing the next color key. The sprite editor provides a command, A, that alleviates this problem by keeping the cursor from advancing after you press a color key. With A activated, you only have to press the cursor-down key to move to the next position.

After you've finished drawing the rectangle, press the X key and notice that

the width of the sprite on the right doubles. Then press the Y key, and the height will double. To save your sprite in memory, press the shift and return keys simultaneously.

When you're asked which sprite you want to edit next, enter 2 and you'll go back to the built-in grid pattern. Then press shift/clear-home to erase the pattern, C to copy another sprite into the box and 1 to designate sprite 1.

The copy command, by the way, is handy for setting up animation. After you draw the first shape and save it, copy it back into the box, make minor changes and save it again, continuing for all eight available sprites. ▶

RUN it right: C-128, in 40-Column mode

## PROGRAMMING

Now press run-stop and request sprite 2 at the prompt, noticing that your copied shape doesn't get saved. Any time you don't want to save a sprite, just press run-stop to return to the prompt.

To access Multicolor mode, press the M key. In this mode, you can draw with three colors, using the 2, 3 and 4 keys, and the cursor becomes a double plus sign. Although you get only half the resolution as in the standard mode, the extra colors will probably compensate. (If not, you can design several single-color sprites that overlap to create the illusion of a multicolor sprite with normal resolution. However, this approach is a waste of the sprites available to you.)

The M command and the A, X and Y commands I mentioned earlier are all toggles; in other words, they're turned on and off by alternately pressing the same key. In Table 1, you'll find a quick-reference list of all the commands available in the sprite editor.

Now draw a multicolor sprite and save it. Then exit the sprite editor by pressing return, instead of a sprite number, at the prompt.

### TURNING ON A SPRITE

Having created and saved a couple of sprites, you can display them on the screen with the Sprite command, which has the following format:

```
SPRITE <number,on/off,color,priority,X-expand,Y-expand,multicolor>
```

The sprite number ranges from 1 to 8 and the color from 1 to 16. The other parameters are turned off and on with values of 0 and 1, respectively. You're probably familiar with all these parameters except priority, which specifies a sprite's location in relation to the

screen data. A value of 0 makes the sprite appear to be in front of the objects on the screen and a value of 1 makes it appear to be behind them.

Type the following line to turn on your first sprite, the rectangle:

```
SPRITE1,1,2,0,0,0
```

After it comes on the screen, try these variations:

```
SPRITE1,1,3,0,1,1 to turn on sprite 1, with red color and X and Y expansion.
```

```
SPRITE1,0,,,0,0 to turn off sprite 1 and cancel X and Y expansion.
```

```
SPRITE2,1,7,0,1,1 to turn on multicolor sprite 2, with blue color and X and Y expansion.
```

Notice the three commas in a row in the second variation. If you want to skip one or more parameters before specifying another one, you must still include all the commas for the computer's reference.

### MOVING A SPRITE

Sprites are positioned and then are moved with two different formats of the MovSpr command. The format for positioning a sprite is:

```
MOVSPR <number,X,Y>
```

The sprite number ranges from 1 to 8. X and Y, which represent the horizontal and vertical coordinates of the upper-left corner of the sprite, range from 0 to 511 and 0 to 255, respectively.

Unlike coordinates on the hi-res screen, not all of these are visible. The corners of the visible area are at 24,50 (upper-left); 344,50 (upper-right); 24,250 (lower-left); and 344,250 (lower-right). As you're placing a sprite, keep in mind that its

boundaries won't be visible against a blank background.

Now, place your second sprite on the screen with:

```
MOVSPR2,150,150
```

The format the MovSpr command takes for setting a sprite in motion is:

```
MOVSPR <sprite number,angle#,speed>
```

The angle, ranging from 0 to 360 degrees, is the direction in which the sprite will move. On the screen, zero degrees is up, 90 degrees is to the right, 180 degrees is down, and so on, just like compass directions on a north-oriented map. The speed can range from 0 to 15, with 15 the fastest. These speeds are fun to play with, and you'll enjoy watching your sprites zip about at speed 15. However, that's really too fast for Basic to handle, and you'll have a hard time keeping the sprites under control.

Type in the following line and watch your sprite move:

```
MOVSPR2,90#7
```

Then try changing the values. When you're done, stop the sprite and reposition it with:

```
MOVSPR2,0#0  
MOVSPR2,150,150
```

Now notice the various colors in the sprite. One is the background color provided by the computer, and you set another in your initial Sprite command. The other two colors can be set with the Basic 7.0 SprColor command. Here's the format:

```
SPRCOLOR<multicolor1,multicolor2>
```

These two colors, along with the background color, will be common for all the sprites on the screen; the only color that can be unique is the one set with the Sprite command.

The following line will produce a red, white and blue sprite:

```
SPRCOLOR3,2
```

Try turning it into a flag and then experimenting with its colors. What would the Star-Spangled Banner look like flying across the screen in green?

### SPRITE COLLISIONS

Since sprites can move, they frequently "collide" with each other and other objects on the screen. Basic 7.0 provides two commands for handling such events, and one is appropriately named Collision. It takes the following format:

```
COLLISION <type,line number>
```

Table 1. Sprite editor commands.

Command	Result
Cursor keys	Move the cursor around the grid.
Home key	Moves the cursor to the top-left corner of the grid.
Shift/clear-home key	Clears the grid and homes the cursor.
Return key	Moves the cursor to the beginning of the next line.
Run-stop key	Exits the current sprite-definition grid.
A	Toggles cursor advance on and off.
1	Erases data "under" the cursor.
2	Places foreground data "under" the cursor.
3 and 4	Place multicolor data "under" the cursor.
Control/1-8	Activate the first eight foreground colors.
Commodore/1-8	Activate the last eight foreground colors.
C	Copies sprite data between two sprites.
M	Toggles Multicolor mode on and off.
X	Toggles horizontal expansion on and off.
Y	Toggles vertical expansion on and off.

## PROGRAMMING

It acts like a Goto command, sending execution to a subroutine at the specified line number when a collision is detected. After the subroutine is done, you can revert to where you left off with a Return command.

The Collision command can handle three types of collisions:

1. Sprite-to-sprite.
2. Sprite-to-character.
3. Light pen.

In this tutorial, I'll deal with only the first two. Sprite-to-sprite is self-explanatory, but note that "character" in sprite-to-character doesn't necessarily mean a letter on the screen; it can also mean a graphics character on the hi-res screen.

The Collision command comes in handy for simple programs, but the slowness of Basic limits its usefulness for the following reason. When a collision is detected, the computer doesn't jump to the subroutine until it finishes executing the current command. During this time, the two sprites that collided can travel a good distance, because the computer automatically moves sprites during the hardware-interrupt interval. Because this interval is too short for Basic to handle, some people turn to the speed of machine language.

The second command available for handling sprite collisions is Bump. It has two possible modes:

1. Sprite-to-sprite.
2. Sprite-to-character.

Bump is difficult to understand at first, because it doesn't come out and say, for instance, that sprites 1 and 2 collided; it reports a 3 instead. Three? Bump treats the eight available sprites as bits in a byte, so they have the usual place values for bits: 1, 2, 4, 8, 16, 32, 64 and 128. The number the Bump command returns is the total of the place values of the sprites that collided. For example, if sprite 1 and sprite 8 collided, a value of 129 would be returned.

Listing 1 contains a short program that stages a sprite race to illustrate simple collision handling.

### Listing 1. Sprite Race program.

```
100 SCNCLR:GRAPHIC
    1,1:BOX1,1,10,319,20,,1
110 SPRITE1,1,3:SPRITE2,1,7,,1:
    SPRCOLOR6,8
120 MOVSPR1,100,200:MOVSPR2,270,200
140 S(1)=INT(RND(1)*5)+1:S(2)=INT
    (RND(1)*5)+1:IFS(1)=S(2)THEN140
150 MOVSPR1,0#S(1):MOVSPR2,0#S(2)
155 COLLISION2:COLLISION2,170
160 GOTO160
```

```
170 A=BUMP(2):PRINTA
175 COLLISION2
180 MOVSPR1,0#0:MOVSPR2,0#0
190 IFBUMP(2)=1THENPRINT"SPRITE
    ONE IS THE WINNER!"
200 IFBUMP(2)=2THENPRINT"SPRITE
    TWO IS THE WINNER!"
210 GRAPHIC0
```

Line 100 draws the finish line for the race. Line 110 turns on your two sprites and sets the multicolor values for sprite 2. Line 120 positions the sprites, line 140 selects a random speed for each and line 150 sets them in motion.

In line 155, notice that the first Collision command has no line number. When the line number is omitted, a Collision command clears any previous collision values. The second Collision command in line 155 sends execution to line 170 when a sprite-to-character collision is detected. Line 160 is just an endless loop that repeats until a collision occurs.

Line 170, which begins the collision-handling subroutine, stores the value returned by the Bump command and prints that value. Line 175 clears all collision information, and line 180 stops the sprites. The computer determines the winner by checking the Bump value again in line 190 and, if necessary, line 200. Finally, the message is displayed by line 210.

The value of Bump is taken twice to illustrate a point. The Bump command is temperamental, and, if you run the program several times, you'll notice that the value stored in variable A isn't always correct for the winner. Sometimes it will be 3, which would mean that both sprites hit the background at the same time. That's why line 175 cancels any previous collision values and checks the values again. This is possible with this program because the sprites are frozen immediately.

Try experimenting with the Sprite Race program by adding more contestants or making other changes.

### ANIMATION

You can do animation on the C-128 with both hi-res graphics and sprites. However, the latter are more effective, because they provide a much more fluid transition between shapes.

Animation is created by rapid "flipping" through a series of pictures, each of which is slightly changed from the previous one. Because our eyes retain an image briefly after the object is gone, we don't detect the gap between pictures, but seem to see the sequence as continuous.

This illusion is easy to create on the C-128 because of the SprSav command, which lets you copy data between sprites and variables. The format for this command is:

SPRSAV <source,destination>

Both source and destination of the data can be either a sprite or a string variable. The latter stores a hi-res shape that has usually been saved with the SShape command. The shape can be recopied to the hi-res screen, or, more important for our purposes, to a sprite. The format is:

SHAPE<variable\$,X,Y,X2,Y2>

The X and Y coordinates mark the top-left corner of the shape, while X2 and Y2 mark the bottom-right. Be careful not to make the shape too big—sprites can't handle shapes bigger than 24x21. You'll avoid any problems if you always use coordinates 0,0,23,20 when saving shapes that will be used with sprites. You'll also avoid problems by making sure you're not in a Multicolor graphics mode.

### IN THE OLYMPIC SPIRIT

Listing 2 contains a small program that shows simple animation: the Olympic rings spinning on their axes. Type in the program and then run it.

#### Listing 2. Olympic Ring program.

```
100 GRAPHIC1,1:COLOR0,1:
    COLOR4,1:COLOR1,2
110 WIDTH2
130 FORT=10TO1STEP-1
140 CIRCLE1,10,10,T,10
150 SSHAPE$(T),0,0,23,20
160 GRAPHIC1,1
170 NEXTT
190 A=1:T=2
200 MOVSPR1,100,100:MOVSPR2,
    115,100:MOVSPR3,130,100
205 MOVSPR4,107,115:MOVSPR5,122,115
210 DO
220 T=T+A:IFT=10ORT=1
    THENA=-A
230 FORB=1TO5
240 SPRSAV$(T),B
250 SPRITEB,1,B+1
260 NEXTB
270 LOOP
```

Line 100 sets up the graphics screen, and line 110 sets the pixel width to double the normal size. Line 130 creates a loop, and line 140 draws a circle that decreases in width with each loop. Line 150 is the key to this loop: It saves the shape drawn in line 140 for use in sprites. Line 160 clears the screen, and line 170 executes the loop until T equals ▶

## PROGRAMMING

1. Line 190 sets variable A, the increment, and variable T, the pointer that indicates which shape to copy into the sprites' data registers. Then lines 200 and 210 position the sprites.

The endless loop in lines 210-270 is the heart of the program; it creates the animation. Within the loop, line 220 updates the shape pointer, and the For statement in line 230 starts an inner loop that copies the shape data into each of the five sprites used. Line 240 does the actual copying, with line 250 turning on the sprites. The operation in line 250 could have been placed in its own loop before line 210, but the program is shorter with it here. Line 260 ends the For-Next loop, while line 270 restarts the main loop.

When you run this program, you'll see five different-colored rings spinning on your screen. They'll continue spinning until you press run-stop/reset, followed by GRAPHIC0. Study the program and perhaps make a few changes, such as substituting the following lines:

```
140 BOX1,10 - T,1,10 + T,20
140 BOX1,10 - T,1,10 + T,20,36*T
140 CIRCLE1,10,10,T,10,,,36*T
```

### SAVING AND LOADING SPRITES

After you've drawn some detailed sprites, save them to disk for later use.

To save your sprites, type:

```
BSAVE"name",B0,P3584TOP4096
```

To reload them later, type in:

```
BLOAD"name",B0
```

### READING SPRITE PARAMETERS

If you're using your sprites in a program where they're constantly changing position, color, and so forth, you may want to find out what their current condition is. Three commands—RSprite, RSpPos and RSpColor—will tell you almost everything you want to know. The formats for these are:

```
RSPRITE <sprite,attribute>
RSPPOS <sprite,characteristic>
RSPCOLOR <register>
```

The sprite parameter is the number of the sprite for which you want the information, and the attribute parameter is the sprite attribute for which you want the value. The attribute values range from 0 to 5 for each of the six parameters available with the Sprite command. An example would be:

```
PRINT RSPRITE (1,1)
```

If a value of 1 was returned, the sprite was on.

Characteristic is the same as attribute, but for different parameters. Characteristic values can be 0 for the X location, 1 for the Y location and 2 for speed. Register values are 1 for multi-color 1 and 2 for multicolor 2.

Try different variations of the programs here. Start off simply, then work your way up, and soon you'll be doing things you never thought possible. ■

*Rob Kennedy is a university student with several years' computing experience under his belt and several magazine articles to his credit.*

# FOOTBALL • BASKETBALL • BASEBALL

**SPORTS FANS...THE SPORTS SIMULATIONS YOU HAVE BEEN WAITING FOR ARE HERE!**

## 3 IN 1 FOOTBALL

- with Stats Compiler for each player and team • you choose from 14 offensive plays and 6 defensive formations • includes 180 college teams and the 28 Pro teams from the '87 season PLUS 174 great college and 189 great pro teams of the past

**NEW!**  
**3-POINT SHOT**

## COURT SIDE COLLEGE BASKETBALL & BASKETBALL: THE PRO GAME

- each player contributes as they did in real life • Stats Compiler • you determine starting lineup, substitutions, shot selection, passing, offensive and defensive styles of play and more • the College game includes 292 teams from the '87-'88 season plus 70 all-time greats
- the Pro game features the 23 Pro teams from '87-'88 and more than 125 great teams of the past

## FULL COUNT BASEBALL

- Includes all 26 teams from the most recent and 52 great teams from the past • 29 man rosters • Ball park effects • Stats Compiler automatically keeps all player and team stats as well as past schedule results. • Complete boxscore to screen and/or printer after each game. • One player vs. computer manager, two-player, and auto-play options. • Input your own teams, draft or trade players from teams already included. • You choose the starting lineups, batting order, relief pitchers, plus game decisions like when to hit away, bunt for a hit, sacrifice, steal, hit & run, bring in the corners or the entire infield, take an extra base, DH option and more!

FULL COUNT Standings & League Leader Program \$14.99

OTHER PAST SEASONS' TEAMS DISKS AVAILABLE AND NEW SEASONS' READY PRIOR TO PLAYOFFS FOR ALL GAMES.

Send check or money order for \$39.99 each. Visa and MasterCard accepted on phone orders only.

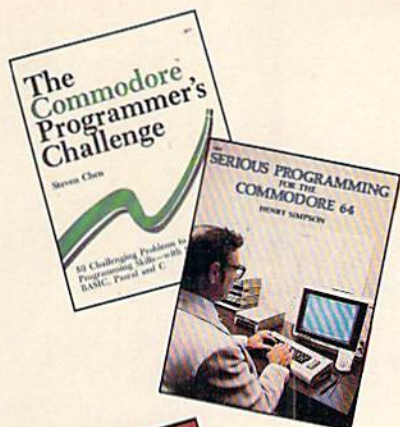
Please add \$2.00 for postage and handling.

**LANCE HAFFNER GAMES**

P.O. Box 100594 • Nashville, TN • 37210 • 615/242-2617

Circle 410 on Reader Service card.

# Discover How Much More You Can Do With Computer Guides From TAB



## THE COMMODORE PROGRAMMER'S CHALLENGE: 50 Challenging Programs to Test Your Programming Skills—With Solutions in BASIC, Pascal, and C

by Steven Chen

Have fun while expanding your programming expertise. These stimulating problems include: mathematical questions, character problems involving sophisticated logic procedures, and applications programs that demand use of your intuition, deductive reasoning skills, and business acumen. For the 64/128. No. 2817P, \$14.95

## SERIOUS PROGRAMMING FOR THE COMMODORE 64™

by Henry Simpson

"... outlines good programming techniques as well as helping you get the most out of the C64"—Rainbo Electronic Reviews

Develop clearly organized, professional-quality programs with the help of this guide. Shows you how to program from the top down using a series of modules and subroutines. No. 1821P, \$10.95

## COMMODORE 128™ BASIC PROGRAMMING TECHNIQUES

by Martin Hardee

Programming in BASIC becomes fast and easy with the help of this expert guide. You'll master telephone communications techniques, storing and retrieving data, and sound and graphics commands. Over 50 programs yield a wide range of applications. No. 2732H, \$18.95

## COMMODORE 128™ DATA FILE PROGRAMMING

by David Miller

Take advantage of the 128's 80-column monitor and other special capabilities with this collection of file-handling techniques and shortcuts. You'll develop a working mailing list database, a mathematics tutorial system, a personal medical records system, a home inventory system, and a stock market portfolio manager. No. 2805H, \$21.95

## TROUBLESHOOTING AND REPAIRING YOUR COMMODORE 64™

by Art Margolis

"With the complete set of schematics and many well placed illustrations, this is an excellent book to help anyone learn to make repairs to the Commodore 64"—Online Today

Symptom analysis charts, step-by-step repair instructions, safety precautions, and your own chip location guide enable you to confidently repair your own C-64. No. 1889H, \$22.95

## ADVANCED COMMODORE 128™ GRAPHICS AND SOUND PROGRAMMING

by Stan Krute

Create high-performance graphics and sound with your C-128. Commodore expert Stan Krute shows you how with the complete source code for two programs. The first is an 80-column graphics package for drawing lines and polygons, pattern painting, and more. The second, a sound and music package, allows you to record, edit, print out, and play back sound compositions. No. 2630H, \$21.95

## ARTIFICIAL INTELLIGENCE PROJECTS FOR THE COMMODORE 64™

by Timothy J. O'Malley

"Well worth the cost and lots of fun!"—Rainbo Electronic Reviews

Explore artificial intelligence with the 16 BASIC programs in this book. They demonstrate tree searches (testing all possible solutions to a problem), heuristics (a modified trial-and-error technique), algorithms, and pattern searching/recognition routines, as well as game and puzzle programs. No. 1883P, \$12.95

## COMMODORE 64™ ADVANCED GAME DESIGN

by George A. and Nancy E. Schwenk

Featuring three full-length games that alone are worth the price, this unique guide shows how you can create exciting games for fun or profit. Using the games as models, it explains: how to develop a game program... what makes a good game... the pros and cons of game programming in BASIC, assembly language, and FORTH... and more. No. 1923P, \$10.95



**TOLL-FREE ORDERING**

**1-800-343-0728**

Ask for the TAB BOOKS operator.

### SATISFACTION GUARANTEED

If you are not completely satisfied with the books you receive, you may return it (them) within 15 days for a complete refund—no questions asked!

YES, I want to get more from my Commodore with these great books from TAB. Send me the following:

No. \_\_\_\_\_ \$ \_\_\_\_\_

No. \_\_\_\_\_ \$ \_\_\_\_\_

No. \_\_\_\_\_ \$ \_\_\_\_\_

Mail coupon to: TAB BOOKS Inc., Blue Ridge Summit, PA 17294-0840

Check/money order enclosed made payable to TAB BOOKS Inc.  
Charge my  VISA  MasterCard  American Express

Acct. No. \_\_\_\_\_ Exp. \_\_\_\_\_

Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State/Zip \_\_\_\_\_

PA residents add 6% sales tax. Orders subject to credit approval. Prices subject to change.

RM108

# The Sound of Basic

*Bells, whistles, siren sounds or beautiful music—let's hear it from your Commodore.*



By BRUCE JAEGER

**W**e Commodore owners sometimes forget how good we have it, especially when it comes to the sound and music capabilities of our machines. What was state-of-the-art sound when introduced by the C-64 is still impressive, and, while the C-128 didn't add another SID chip to give us six voices, it did give us some Basic sound and music commands that make working with sound a lot easier.

## A SID PRIMER

Just like the old-time fiddler who said, "Sure, I know how to read music—but not enough to hurt my playing," you don't have to memorize everything there is to know about the SID chip to get great sounds and music out of your C-64 or C-128. However, a little knowledge will keep your experiments going in the right direction.

All sounds on the C-64 or C-128 are produced by the MOS 6581 SID (sound interface device). This chip has three independent voices that can make three different sounds at once. The SID is controlled by placing different values (whole bytes, or sometimes just certain bits in a byte) in specified memory locations called registers.

With the C-64, you have to place these values yourself, using memory-store operations from a machine language program or Poke commands from Basic. The Sound, Envelope, Volume and Play commands in the C-128's Basic 7.0 do these Pokes automatically, and I'll give you some examples that show how easily some very complex sound effects can be achieved. However, you can still create sounds the 64 way, as long as you're in bank 15. So, nothing you learn about sound on the C-64 has to go to waste with the C-128.

Be sure to read pages 457-469 in the *C-64 Programmer's Reference Guide* or pages 605-610 in the *C-128 Programmer's*

*Reference Guide* for a detailed description of the SID chip and its control registers. Even if you're a C-128 owner, you'll probably want to buy the C-64 book; the 128 version is pretty skimpy on SID.

## CHARACTERISTICS OF SOUNDS

A note's **frequency** is the number of cycles per second of its waveform, which produces the pitch of the note—in other words, how high or low the note sounds. A note with a greater frequency (or pitch) sounds higher than one with a lower frequency.

The pitch of each of the SID's three voices is specified by poking values into a two-byte register, in the low-byte, high-byte style familiar to 6502 programmers. Using two bytes is necessary because each byte can hold values only from 0 to 255, while frequency needs to range up into the thousands. The value stored in the high-byte register (Freq Hi) is multiplied by 256 and added to the number in the low-byte register (Freq Lo). For example, to store the number 440 in the frequency registers, you'd poke 1 into the high byte and 184 into the low byte:  $1 * 256 + 184 = 440$ .

Unfortunately, these values don't correspond to actual frequencies; 440 poked into a SID frequency register doesn't produce the 440-cycle-per-second (Hertz) A note. So, Commodore has included frequency conversion tables for your use in both the C-64 and C-128 programming guides.

## HARMONICS

We say a 440 Hz A note has a fundamental frequency of 440 Hz. That frequency is also called the **first harmonic**, because it's the most important component of the sound. However, on just about any instrument other than the purest of sine-wave generators you'll hear additional harmonics: a second harmonic of 880 Hz (twice that of the

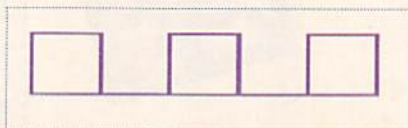
fundamental frequency), a third harmonic of 1320 Hz (three times the fundamental frequency), and so on.

The harmonics often get weaker as they increase in frequency. In fact, most of the time you're not really aware of any frequencies but the fundamental one until you remove the harmonics and hear how "thin" the sound becomes. Harmonics are just one of the reasons the same note sounds differently when played on a trumpet, violin or clarinet.

## WAVEFORMS

Commodore has used different mathematical combinations of harmonics to create three of the four **waveforms** the SID chip can produce: Pulse, Triangle and Sawtooth. The fourth waveform, Noise, was carefully designed to have no mathematically predictable sound at all!

Pretend there's a sound with a frequency of one vibration per second making your eardrums move. This frequency is far too low to actually hear, of course, but it will be easy to use for visualizing waveforms. Suppose the sound wave hits your right ear, instantaneously moves your eardrum to the left, waits a half-second, then instantaneously stops, releasing the eardrum to the right for the remaining half-second. A graphic representation of this "pulse" or "square wave" waveform might look like this:

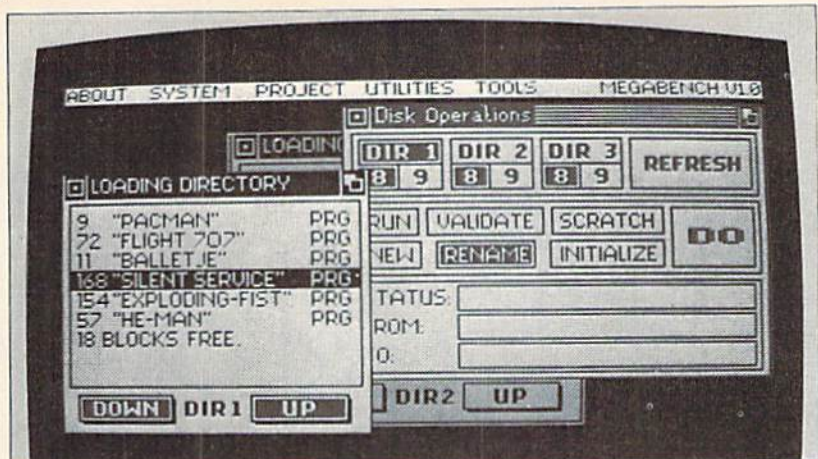


**Figure 1. Graphic representation of the Pulse waveform.**

(This is a theoretically perfect waveform; never mind for now the inertia of your eardrum, or of molecules, that ▶

NEW IMPROVED

NEW IMPROVED



# FINAL CARTRIDGE III®

The Best Utility Ever for Your C-64 or 128

Only! **\$69.95** Only!

**SPECIAL VALUE: FREE Joystick and FREE 100 PARAMETER PAK,**  
Total Retail Value, **\$39.95**, with each purchase **FINAL CARTRIDGE III**

This powerful ROM-based operating system contains easy-to-use windows and pull down menus. Allowing the user to select either mouse, joystick or keyboard, he may access over 60 new commands and functions. Let your C-64 perform like an Amiga. Various printer interfaces as well as a basic toolkit can also be accessed.

**STATE OF THE ART FREEZER**  
Includes variable size screen dumps (color if Epson color or NEC is used). Allows total backup of any memory-resident software on the market today! Files are packed and reloadable without the cartridge, 60K in just 15 seconds. Exits to Basic or ML monitor.

**EXTENDED ML MONITOR**  
Does not reside in memory! Includes 1541 drive access and sprite editing. Features up and down scrolling and printer driver!

**NOTEPAD/WORD PROCESSOR**  
Contains proportional characters and word wrap. Enables you to store and print small notes, etc.

**FASTEST DISK LOADING EVER!**  
Contains 2 disk loaders, with speeds up to 15x faster than normal!

**TRANSFORM YOUR C-64 INTO AN AMIGA LOOK-ALIKE!**  
Various windows such as: Preferences, Tape, Disk Windows, Directory, Printer & Clock allow you to feel as if you are working in the same friendly environment as the Amiga!

**EASY-TO-USE MENU BAR**  
Almost any command not activated by windows can be accessed while in Basic by just typing in Box.

**BASIC TOOLKIT & KEYBOARD EXTRAS**  
Includes: Renumbering, auto, old, delete, kill, save, 24K RAM for Basic, fast format and many, many more.

*"...I can't begin to think of a cartridge which does so many useful things. . . a tremendous value, a must item for the BASIC and machine language programmer."*  
—Art Hunkins, Compute's Gazette 7/87



**GAMES KILLER**  
Kills sprite to sprite and/or background collision. Can be started at any point in your games.

**AUTO FIRE ENGINE**  
Transforms normal joystick into an auto fire!

**EASY-TO-USE RESET SYSTEM**  
Reset your computer by the simple touch of a button!! Wow!!

*"No need for all those extras when you have this C-64 assistant. . . a conventional review doesn't do the Final Cartridge justice. . . fun at a price is a rarity."*

—Tim Walsh, RUN Magazine 9/87



Home & Personal Computers of America  
154 Valley Street  
South Orange, NJ 07079  
201-763-3946, dealers only, 201-763-1693

## INTRODUCING SUPER CARD

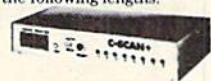
Backs up any software program! Even the latest protection schemes! Plugs into your drive with only the use of a screwdriver. If anything could back up everything, this is it. 100% satisfaction guaranteed! 10 day or money back guarantee!

**ONLY \$39.95!**

**FINAL CARTRIDGE II**  
**ONLY \$24.95**

Call or write for more information

**Attention Schools and Educators:**  
C-Scan+ is the ultimate network for Commodore computers, eight computers share one or two disk drives, and only one printer and software program is needed. Simple installation, auto scanning and auto power on. Works perfectly with The Final Cartridge. 1 year warranty.  
C-Scan+ . . . . . **\$199.95**  
Cables available in the following lengths:  
9 ft. . . \$13.95  
12 ft. . . \$15.95  
18 ft. . . \$17.95  
36 ft. . . \$19.95



## EXTRAS AVAILABLE

- \*Final Cartridge I. . . . . \$14.95
  - \*C-1351 Mouse . . . . . \$32.95
  - Deluxe Joystick . . . . . \$ 8.95
  - Cent. printer cable\* (optional) . . . . . \$19.95
- \* Limited quantities available.

**Ordering Info:**  
**Orders Only Call:**  
**1-800-458-8682**  
MC/Visa accepted. Money orders (immediate shipments). Personal check (allow 2-3 weeks for check clearing). NY & NJ residents add appropriate sales tax. Add \$3.00 for s/h. Questions and info. call:  
(201) 763-3946  
Fax: (201) 763-1693  
**ANY PRODUCT PURCHASED FROM DATEL ELECTRONICS WILL NOT BE GUARANTEED BY H&P COMPUTER.**

## PROGRAMMING

makes a perfectly square waveform impossible.)

Now, suppose the sound wave hits your ear, gradually and evenly moves your eardrum to the left for a half-second, then gradually moves the eardrum back to the right. A graphic representation of this waveform is called a "triangle," because it looks like this:

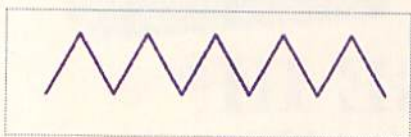


Figure 2. The Triangle waveform.

Next, suppose the sound wave hits your ear, gradually and evenly moves your eardrum to the left for a second, then stops, instantly releasing the eardrum back to the right. A graphic representation of this waveform is called a "sawtooth":

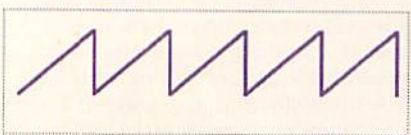


Figure 3. The Sawtooth waveform.

Finally, suppose the sound wave hits your eardrum with all sorts of random pulses. This Noise waveform might look like this:



Figure 4. The Noise waveform.

The C-64 program in Listing 1 demonstrates what each of these waveforms sounds like. You can run this program on a C-128 also, but precede it with a BANK 15 command. (Don't worry for now if you don't understand what all the Peek and Poke commands mean; I'll explain them later.)

The program in Listing 2 does the same thing, but it's designed for the C-128. Notice how much simpler it is.

Listing 1. Waveform comparison program [C-64].

```
10 REM WAVEFORMS (C64) :REM*249
20 SD=54272:PRINT CHR$(147)
:REM*143
```

```
30 GOSUB 150:REM RESET SID
:REM*199
40 POKE SD+2,100:POKE SD+3,1
:REM*232
50 POKE SD+24,15 :REM*165
60 POKE SD+6,240 :REM*186
70 POKE SD+1,50:POKE SD,1
:REM*159
80 FOR X=1 TO 4 :REM*207
90 READ WV,WV$:PRINT WV$
:REM*250
100 POKE SD+4,WV :REM*62
110 POKE SD+4,PEEK(SD+4) OR 1:R
EM START A/D/S CYCLE
:REM*247
120 FOR DELAY=1 TO 1000:NEXT DE
LAY :REM*45
130 NEXT X:GOSUB 150:END
:REM*103
140 : :REM*198
150 FOR X=0 TO 23:POKE SD+X,0:N
EXT:RETURN :REM*119
160 : :REM*218
170 REM WAVEFORM DATA :REM*6
180 DATA 16,TRIANGLE:REM BIT 4
:REM*198
190 DATA 32,SAWTOOTH:REM BIT 5
:REM*88
200 DATA 64,PULSE:REM BIT 6
:REM*134
210 DATA 128,NOISE:REM BIT 7
:REM*249
```

Listing 2. Waveform comparison program [C-128].

```
10 REM WAVEFORMS (C128):REM*216
20 PRINT CHR$(147):VOL 15
:REM*140
30 FOR X=1 TO 4 :REM*157
40 READ WV,WV$:PRINT WV$
:REM*172
50 SOUND 1,12000,35,0,0,0,WV,20
48 :REM*173
60 SLEEP 1:NEXT X :REM*98
70 VOL 0:END :REM*43
80 : :REM*138
90 REM WAVEFORM DATA :REM*215
100 DATA 0,TRIANGLE :REM*3
110 DATA 1,SAWTOOTH :REM*73
120 DATA 2,PULSE :REM*10
130 DATA 3,NOISE :REM*223
```

### VOLUME

The volume, or amplitude, of a sound is merely how loud it is. Let's say we have a sound with a Sawtooth waveform that looks like this:



Figure 5. A sound created with the Sawtooth waveform.

The same note, louder, might appear like this:

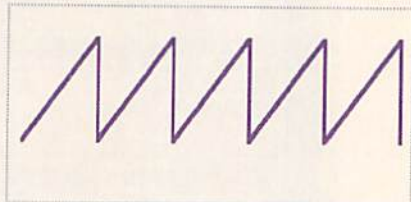


Figure 6. The same sound, only louder.

Note that the distances between the waves are the same in each case, so the two notes are of the same frequency. However, the amplitude of the second note is greater, so the note is louder.

### ENVELOPE

The envelope of a note includes four different components of a note's tone: **attack**, or the time it takes the note to reach its peak volume; **decay**, or how soon it drops down to what might be called the note's average volume level; **sustain**, or how long it stays at that level; and, finally, **release**, or how long it takes for the sound to stop. Here's a graphic representation of how an envelope might look:

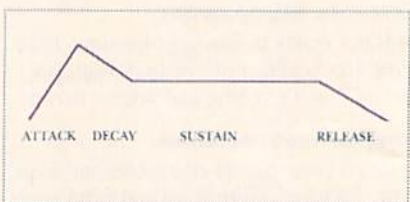


Figure 7. Representation of a sound's envelope.

Note that, unlike the previous diagrams, which showed a sound's frequency, this graph shows the relative volume, from when the note begins to when it dies off. If the sound were that of a hammer hitting a length of two-by-four, the attack and decay part of the envelope would be accentuated, with almost no sustain or release, because hammers and two-by-fours don't resonate very well. When a bow is drawn across a violin string, however, the attack is gentler, with almost no decay, and the sound sustains until the bow stops moving, at which point it releases at a rate that says much about the quality of the violin.

Listing 3 is a C-64 program you can use to play with setting rates of attack, decay, sustain and release. Try the values listed in Figure 8 and note the different results. The program will keep



## PROGRAMMING

running until you input a negative number for Attack.

### Listing 3. Envelope experiment program (C-64).

```

10 REM ENVELOPES (C64) :REM*174
20 SD=54272:PRINT CHR$(147) :REM*143
30 GOSUB 240:REM RESET SID :REM*200
40 POKE SD+24,15 :REM*175
50 INPUT "ATTACK (0-15) ";A :REM*96
60 IF A<0 THEN END :REM*111
70 INPUT "DECAY (0-15) ";D :REM*122
80 INPUT "SUSTAIN (0-15) ";S :REM*40
90 INPUT "RELEASE (0-15) ";R :REM*22
100 AD=(A*16)+D :REM*47
110 SR=(S*16)+R :REM*124
120 POKE SD+5,AD:REM ATTACK/DECAY :REM*97
130 POKE SD+6,SR:REM SUSTAIN/RELEASE :REM*149
140 POKE SD+1,25:POKE SD,1:REM FREQ :REM*78
150 POKE SD+4,32:REM SAWTOOTH WAVEFORM :REM*163
160 : :REM*218
170 POKE SD+4,PEEK(SD+4) OR 1:REM EM START A/D/S :REM*13
180 FOR DELAY=1 TO 1000:NEXT :REM*147
190 POKE SD+4,PEEK(SD+4) AND 254:REM RELEASE :REM*10
200 : :REM*3
210 FOR DELAY=1 TO 1000:NEXT :REM*245
220 RUN :REM*103
230 : :REM*33
240 FOR X=0 TO 23:POKE SD+X,0:NEXT:RETURN :REM*201
    
```

The C-128 has a built-in Envelope command for easily setting attack, decay, sustain and release, as well as for choosing waveforms and other things. Unfortunately, envelopes created with this command don't work with the C-128's Sound command; you can use them only with the Play command.

Besides letting you design your own envelopes, the C-128 comes with ten predefined envelopes for use with the Play command. Listing 4 demonstrates these built-in envelopes, which are supposed to sound like various musical instruments.

### Listing 4. Built-in envelopes program for the C-128.

```

10 REM ENVELOPES (C128):REM*231
20 PRINT CHR$(147) :REM*226
30 FOR X=0 TO 9 :REM*167
40 READ E$:PRINT E$ :REM*44
50 PLAY "T"+STR$(X)+"03 CDEFGAB
    
```

```

04 C" :REM*204
60 PRINT:SLEEP 1:NEXT X :REM*3
70 : :REM*128
80 DATA PIANO,ACCORDIAN,CALLIOP E,DRUM,FLUTE :REM*165
90 DATA GUITAR,HARPSICHORD,ORGAN,TRUMPET :REM*38
100 DATA XYLOPHONE :REM*12
    
```

You can use the Envelope command to redefine any of the built-in envelopes. The format is:

ENVELOPE number, attack, decay, sustain, release, waveform, pulse width

Number is the envelope number (0-9) you want to modify; attack, decay, sustain, and release have values from 0 to 15; waveform ranges from 0 to 3 (look at the Data statements in the C-128 Waveform program); and pulse width is a value from 0 to 4095 that affects only the Pulse waveform.

### Listing 5. Defining envelopes program for the C-128.

```

10 REM ENVELOPE #,ATTACK,DECAY,SUSTAIN,RELEASE,WAVEFORM,PULSE WIDTH :REM*152
20 ENVELOPE 1,8,2,2,2,1,2000 :REM*122
30 PLAY "T1 03 CDEFGAB 04 C" :REM*54
    
```

Listing 5 is a quick program you can use to try defining envelopes on the

C-128. Run the program over and over, varying the values in line 20 (except for the first 1).

## FILTERS

The SID can use filters to remove certain ranges of frequencies from a sound, to help you achieve the sound you want. Three types of filters are available: low-pass, high-pass and band-pass. To use them, you must set frequency cutoff points. A low-pass filter lets any frequency below the cutoff point through, so you hear just lower harmonics. A high-pass filter lets any frequency above the cutoff point through, so you hear just higher harmonics. With a band-pass filter, you set both high-frequency and low-frequency cutoff points, spaced so that the frequencies you desire would "fit" between them.

For more information on filters, refer to the section on filtering in the C-64 programming guide (page 199) and to the description of the Filter command in the C-128 guide.

## CREATING SOUNDS

Twenty-four bytes control the sounds coming out of the SID chip. These start at memory address 54272, to which variable SD is set in all the example programs in this article. The parameters these bytes define are listed in Table 1. For more detailed information on the SID-controlling bytes, refer to discus-

Figure 8. Trial values for program in Listing 3.

```

Attack = 10 : Decay = 0 : Sustain = 0 : Release = 0
Attack = 0 : Decay = 10 : Sustain = 0 : Release = 0
Attack = 10 : Decay = 10 : Sustain = 10 : Release = 10
Attack = 15 : Decay = 1 : Sustain = 3 : Release = 3
    
```

Table 1. SID control bytes.

Voice 1	Voice 2	Voice 3
SD low frequency	SD + 7 low frequency	SD + 14 low frequency
SD + 1 high frequency	SD + 8 high frequency	SD + 15 high frequency
SD + 2 pulse data l	SD + 9 pulse data l	SD + 16 pulse data l
SD + 3 pulse data h	SD + 10 pulse data h	SD + 17 pulse data h
SD + 4 control register	SD + 11 control register	SD + 18 control register
SD + 5 attack/decay	SD + 12 attack/decay	SD + 19 attack/decay
SD + 6 sustain/release	SD + 13 sustain/release	SD + 20 sustain/release

```

SD + 21 filter cutoff frequency (low nibble)
SD + 22 filter cutoff frequency (high byte)
SD + 23 filter resonance control
    
```

SD + 24 Volume Control, Filter Control

Note: SD stands for memory address 54272.

## PROGRAMMING

sions of SID and the memory map in the C-64 programming guide.

### BITS?

The Voice 1 Control register and the Volume and Filter Control registers are the most confusing, because individual bits, instead of bytes, control the SID. This isn't the place for a tutorial on the binary number system, but a little bit about it is necessary. Our familiar decimal system, based on the powers of 10, uses ten numerals, 0-9, with the actual value of each numeral in a number depending on its "place" in the number; e.g., units, tens, hundreds, etc. In the binary system, based on the powers of 2, there are just two numerals: 0 and 1. We use binary in computers because 0 can stand for "off" and 1 for "on," which lends itself nicely to electronic switching.

Every byte is made up of eight bits, with each bit either 1 (on) or 0 (off), depending on whether or not the bit "place value" is used in the number.

Figure 9 shows the decimal value of each bit in the following binary number: 1 0 0 1 0 0 1 1. There's a 1 in the "128" column, a 1 in the "16" column, a 1 in the "2" column and a 1 in the "1" column.  $128 + 16 + 2 + 1 = 147$ . Therefore, 10010011 in binary equals 147 in decimal.

Now, let's see why it's important to know about bits. Refer to Table 2 for a list of the bits controlled by the Control register for Voice 1 and Table 3 for the Volume Control, Filter Control register.

Basic has no direct way of turning a bit on or off. That must be done through And, Or, Peek or Poke commands. (Look up And and Or in your Basic references.) Let's say you want to set bit 5 of the Control register to turn on the Sawtooth waveform. You know that binary bit 5 has a decimal value of 32 (215; check the binary table again), but you can't just poke a 32 into the register, because 32 in binary is 00100000, and all those 0s might turn off something you want on. So, to turn bit 5 on, you

have to peek the value in the Control register, perform a logical Or operation with that value and 32, then poke the value back into the Control register. To turn bit 5 off, you'd peek the value in the Control register, perform an And operation with that number and the result of  $255 - 32$ , and poke the new value back into the Control register. (If that doesn't make sense, write down the binary equivalents of the numbers on paper, and then try it again.)

### PLAY TIME

Don't worry if some of that went over your head; you can pick up the theory gradually while you play with variable values in other people's sound routines. Listings 6-9 provide a few such routines to start with.

#### Listing 6. C-64 siren program.

```

10 REM SIREN (C64) :REM*127
20 SD=54272 :REM*19
30 GOSUB 370:REM ENSURE ALL REG
   ITERS RESET :REM*23
40 POKE SD+3,1:REM SET HIGH PAR
   T OF PULSE WAVEFORM :REM*194
50 POKE SD+2,240:REM SET LOW PA
   RT OF PULSE WAVEFORM :REM*19
60 POKE SD+24,15:REM VOLUME FUL
   L, FILTERS ALL OFF :REM*73
70 REM SET ATTACK TO 1, DECAY T
   O 1 :REM*147
80 POKE SD+5,(1*16)+1:REM ATTAC
   K/DECAY :REM*182
90 REM SET SUSTAIN TO 15, RELEA
   SE TO 1 :REM*217
100 POKE SD+6,(15*16)+1:REM SUS
   TAIN/RELEASE :REM*103
110 POKE SD+4,64:REM CHOOSE PUL
   SE WAVEFORM :REM*120
120 : :REM*178
130 REM START ATTACK/DECAY/SUST
   AIN SEQUENCE IN CONTROL REG
   ISTER :REM*33
140 REM (BY TURNING ON BIT 1)
   :REM*80
150 POKE SD+4,PEEK(SD+4) OR 1
   :REM*237
160 : :REM*218
170 REM SIREN UP :REM*14
180 FOR X=15 TO 85 :REM*220
190 POKE SD,1:REM SOUND LOW FRE
   Q BYTE :REM*150
200 POKE SD+1,X:REM HI FREQ / W
   E'RE CHANGING IT :REM*195
210 GOSUB 340:REM DELAY ROUTINE
   :REM*155
220 NEXT X :REM*12
230 : :REM*33
240 REM SIREN DOWN :REM*245
250 FOR X=84 TO 10 STEP -1
   :REM*149
260 POKE SD,1:REM SOUND LOW FRE
   Q BYTE :REM*93
270 POKE SD+1,X:REM HI FREQ BYT
   E / WE'RE CHANGING IT

```

Figure 9. Breakdown of a sample byte.

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
(128)	(64)	(32)	(16)	(8)	(4)	(2)	(1)
1	0	0	1	0	0	1	1

Table 2. Values in SID Voice 1 Control register.

Bit	Controls...
0	If 1, starts Attack/Decay/Sustain cycle If 0, starts Release cycle
1	If 1, synchronizes Voice 1 oscillator with Voice 3
2	If 1, ring modulates Voice 1 oscillator with Voice 3
3	If 1, disables Voice 1 If 0, enables Voice 1
4	If 1, selects Triangle waveform
5	If 1, selects Sawtooth waveform
6	If 1, selects Pulse waveform
7	If 1, selects Noise waveform

Table 3. Values in SID Volume Control, Filter Control register.

Bit	Controls...
0-3	Control volume, which can be set to any value from 0 to 15. (Go back to the binary chart and add up the possible values of bits 0, 1, 2 and 3. That's $1 + 2 + 4 + 8 = 15$ , the highest possible number that can be represented with three binary bits.)
4	If 1, selects the Filter Low-Pass mode.
5	If 1, selects Filter Band-Pass mode.
6	If 1, selects Filter High-Pass mode.
7	If 1, turns off Voice 3 output

# PROGRAMMING

```

                :REM*128
280 GOSUB 340:REM DELAY ROUTINE
                :REM*193
290 NEXT X      :REM*86
300 :          :REM*103
310 GOSUB 370:REM RESET ALL REG
    ISTERs (SOUND OFF) :REM*205
320 END        :REM*193
330 :          :REM*133
340 REM DELAY LOOP      :REM*42
350 FOR DE=1 TO 2:NEXT DE:RETUR
    N                :REM*58
360 :          :REM*163
370 REM RESET ALL REGISTERs (EX
    CEPT VOLUME)     :REM*127
380 FOR X=0 TO 23:POKE SD+X,0:N
    EXT:RETURN       :REM*92
    
```

For Listing 6, a C-64 program that generates a siren sound, I'll explain the meaning of every operation:

Line 30 makes sure all SID registers are cleared by calling a subroutine that pokes a value of 0 into each register location except the Volume Control byte at SD+24. (I've found that it's the Pokes into SD+24 that cause all the nasty clicks some Basic sound programs have!)

Lines 40 and 50 poke values into SD+2 and SD+3 for use by the Pulse waveform, which is set later in the routine. You can poke any number from 0 to 255 into SD+2 and any number from 0 to 15 into SD+3. I just played around with these numbers until I found a combination I liked.

Line 60 sets the volume to full (15), and the rest of the 0s in the binary representation of 15 (00001111) turn off all the filtering options, which I didn't want. You can do pokes like this directly—without performing And or Or operations with values previously in the register—if you know exactly what you want the whole byte to be.

Line 80 sets Attack and Decay for Voice 1. The upper four bits (xxxx...) of the byte at SD+5 set the attack time (lower numbers are faster), and the lower four bits (...xxxx) set the decay time. (Again, lower numbers are faster.) Half of a byte is called a nibble, by the way. To set Attack in the high nibble, you've got to take your 0-15 number and multiply it by 16, then add the product to your 0-15 value for Decay and poke the sum into SD+5. Commodore surely made this tedious...

Line 100 does the same thing for Sustain and Release, only this time with SD+6.

Line 110 chooses the Pulse waveform by setting bit 6 of the Control register at SD+4. Again, I didn't bother to Or this value into SD+4, because I wanted the rest of the bits set to 0.

Note: Be sure to set the Waveform register (SD+4) just before turning on the sound (see line 150); something about the SID chip makes it "forget" the contents of that register. I don't know why, but I've noticed that this happens.

Line 150 starts things humming. SID can't make any sound at all until bit 1 of the Control register is turned on, to start the attack/decay/sustain cycle. I had to be careful here not to just poke a 1 into SD+4, because that would have put a 0 in the bit 6 that I just set in line 110! So, I did an Or operation with the value resulting from peeking SD+4 with the number 1, then placed the result back into SD+4. (For sounds that have to be "released" to sound right, you'd turn bit 1 off with POKE SD+4, PEEK(SD+4) AND 254. The siren sound in my program doesn't need to be released, so I didn't include a release in the code.)

Even though bit 1 of the Control register has now been set, there's still no sound, because the values in the Voice 1 SD and SD+1 frequency registers are still 0. (A frequency of 0 is very quiet!) This is changed in the Siren Up routine, which raises the sound, and the Siren Down routine, where the sound "swoops" down and stops.

Line 190 places a 1 in the frequency low byte, and line 200 places a changing value (X) in the frequency high byte. Every time the program goes through the two For-Next loops, the value in the high-byte register changes by 1, which is the same as changing the entire frequency value by 256. (Remember, the high byte is multiplied by 256, then added to the low byte, to yield the frequency number.)

Lines 210 and 280 just call a delay loop to stretch out the sound.

After you type in Listing 1 and save it, experiment by changing the values

of the variables. Perhaps use a different waveform in line 110 or different levels of attack, decay, and such. You can also add one or two more voices to the siren by using the appropriate addresses to set up the sound registers in Voice 2 and/or Voice 3.

## THE C-128 SOUND COMMAND

The three little siren programs (Listings 7, 8 and 9) take advantage of the C-128's powerful Sound command, which can do in a line or two what requires a whole program on the C-64.

### Listing 7. C-128 siren program #1.

```

10 REM SIREN (C128) #1 :REM*35
20 SOUND 1,15000,100,0,5000,100
    ,2 :REM*35
30 SOUND 1,15000,100,1,5000,100
    ,2 :REM*11
    
```

### Listing 8. C-128 siren program #2.

```

10 REM SIREN (C128) #2 :REM*85
20 SOUND 1,18000,135,0,4000,100
    ,2 :REM*71
30 SOUND 2,18000,135,0,4000,100
    ,2 :REM*187
40 SOUND 1,18000,135,1,4000,100
    ,2 :REM*161
50 SOUND 2,18300,135,1,4300,100
    ,2 :REM*208
    
```

### Listing 9. C-128 siren program #3.

```

10 REM SIREN (C128) #3 :REM*71
20 SOUND 1,18000,135,0,4000,100
    ,2 :REM*71
30 SOUND 3,20000,235,2,10000,80
    0,1 :REM*247
40 SOUND 1,18000,175,1,400,100,
    1 :REM*60
    
```

The Sound command has the following parameters: voice, frequency, duration, step direction, minimum sweep ▶

Table 4. Parameters of C-128 Sound command.

Voice	1-3
Frequency	0-65535
Duration	0-32767
Direction	0-2. 0 (the default)—sweeps the frequency upward as the sound continues; 1—sweeps the frequency lower as the sound continues; 2—oscillates
Minimum sweep frequency	Directs SID not to sweep below this value (default,0.)
Sweep value	Amount to sweep by; 0 means not to sweep at all, but play a steady frequency
Waveform	0-3. 0—Triangle; 1—Sawtooth; 2—Pulse; 3—Noise
Pulse width	0-4095 (For the Pulse waveform only)

**RUN Works: A One-Disk Software System for Everything Commodore 64 and 128 Users Need**

1. **RUN PAINT** Full-Feature Paint and Drawing Program
2. **MONEY MANAGER** for Business and Home
3. **LABEL BASE** Create Address Labels
4. **RUN TERM** Telecommunicator
5. **RUN SHELL** Disk Utility
6. **GRAPHMAKER** 3-D Bar Graphs
7. **FORM WRITER** Forms Design

*Spend a little and get the works...*

# **RUN WORKS.**

As a home-based business owner, I save time and money with LABEL BASE's fast, easy address labeling system. And I really appreciate FORMWRITER'S form creation program when I think of the money I'd spend creating and printing forms professionally.

When I create a proposal for work, GRAPHMAKER'S 3-D Bar Graphs really help me make my point.

The MONEY MANAGER really lets our family plan our finances and save!

I think RUNPAINT is awesome 'cause I can draw on the screen just by moving the pointer with my joystick or mouse. It's easy!

RUNPAINT lets me design and print my own unique creations. Even though I'm not an artist, RUNPAINT makes me look like one!

**Introducing RUN Works.** . . a complete selection of all the software programs you'll ever need.

**On just one disk!**

**RUN Works is easy to use.** But it works hard so you don't have to. Which means you're more productive and efficient.

And you can buy RUN Works at a fraction of the price you'd pay for comparable programs—up to \$50 each elsewhere.

What's more, RUN Works and its fully illustrated documentation booklet are only available through this special offer.

**So order today.** There's no risk. RUN Works is 100% Money Back Guaranteed for thirty days.

**Call 1-800-343-0728**  
Or send back the coupon or order card today.



**YES!** I want to spend just a little and get the software works for my Commodore 64 or 128. Please rush me all seven RUN Works programs on just one easy-to-use disk.

**I'll pay only \$24.97!**

- Check is enclosed  MasterCard  
 American Express  Visa

CARD# \_\_\_\_\_ EXP. DATE \_\_\_\_\_  
NAME \_\_\_\_\_  
ADDRESS \_\_\_\_\_  
CITY \_\_\_\_\_ STATE \_\_\_\_\_ ZIP \_\_\_\_\_

Foreign Airmail, please add \$3.95 per order.  
Mail this coupon or the postage-paid card to:  
IDG Communications/Peterborough  
Attn: RUN Works RWRS8  
PO Box 802, Peterborough, NH 03458

## PROGRAMMING

frequency, sweep step value, waveform and pulse width.

The possible values for the various parameters are listed in Table 4.

Examining the values I used in the C-128 sirens should give you a good idea how these values work.

### BACK TO NATURE

The C-64 program in Listing 10 simulates the sound of a cricket. To transform the cricket into a frog, make the changes indicated below the listing.

#### Listing 10. Cricket program [C-64].

```

1Ø REM CRICKET (C64)      :REM*191
2Ø SD=54272              :REM*19
3Ø GOSUB 23Ø:REM RESET SID
                          :REM*2Ø3
4Ø POKE SD+24,15:REM FULL VOLUM
E                          :REM*115
5Ø :                      :REM*1Ø8
6Ø GOSUB 11Ø:REM MAKE SOUND
                          :REM*1Ø8
7Ø DT=2Ø:GOSUB 2ØØ:REM DELAY
                          :REM*3Ø
8Ø GOSUB 11Ø:REM SOUND AGAIN
                          :REM*52
9Ø DT=1ØØØ:GOSUB 2ØØ:RUN:REM*24
1ØØ :                    :REM*158
11Ø REM ONE "CRICK" SOUND
                          :REM*95
12Ø FOR X=1 TO 2          :REM*235
13Ø POKE SD+1,254:REM FREQUENCY
                          :REM*135
14Ø POKE SD+4,16:REM TRIANGLE W
AVEFORM                    :REM*243
15Ø POKE SD+4,PEEK(SD+4) OR 1:R
EM START SOUND            :REM*65
16Ø DT=5Ø:GOSUB 2ØØ:REM DELAY
                          :REM*184
17Ø POKE SD+4,PEEK(SD+4) AND 25
4:REM RELEASE             :REM*62
18Ø NEXT:RETURN          :REM*87
19Ø :                    :REM*248
2ØØ REM DELAY ROUTINE    :REM*156
21Ø FOR DE=1 TO DT:NEXT:RETURN
                          :REM*225
22Ø :                    :REM*23
23Ø REM RESET ROUTINE    :REM*2Ø8
24Ø FOR X=Ø TO 23:POKE SD+X,Ø:N
EXT:RETURN                :REM*2Ø1
    
```

For a frog sound:

Change line 130 to read: POKE SD + 1, 20

Add line 135: POKE SD, X\*2

Change line 160 to read: DT = 30 : GOSUB 200

For a change of scene, listen to the surf sound generated by the program in Listing 11.

#### Listing 11. Sound of surf program [C-64].

```

1Ø REM SURF 64          :REM*147
2Ø SD=54272            :REM*19
    
```

```

3Ø GOSUB 17Ø:REM RESET SID
                          :REM*193
4Ø POKE SD+24,15:REM VOLUME
                          :REM*142
5Ø POKE SD+Ø,Ø:REM FREQ LOW
                          :REM*252
6Ø POKE SD+1,2ØØ:REM FREQ HIGH
                          :REM*176
7Ø A=1Ø:REM ATTACK=1Ø   :REM*187
8Ø D=12:REM DECAY=12    :REM*188
9Ø POKE SD+5,(A*16)+D:REM SET A
TTACK/DECAY              :REM*7Ø
1ØØ POKE SD+4,128:REM SET NOISE
WAVEFORM                 :REM*9Ø
11Ø POKE SD+4,PEEK(SD+4) OR 1:R
EM START                 :REM*13Ø
12Ø REM WAIT FOR SOUND TO FINIS
H                         :REM*88
13Ø FOR DELAY=1 TO 2ØØØ:NEXT DE
LAY                       :REM*193
14Ø GOTO 5Ø:REM REPEAT  :REM*176
15Ø END                  :REM*23
16Ø :                    :REM*218
17Ø REM RESET SID      :REM*52
18Ø FOR X=1 TO 23:POKE SD+X,Ø:N
EXT:RETURN               :REM*17
    
```

Then, to have Fourth of July at the beach, turn the waves into explosions by changing the Attack variable, A, in line 70 to 3. For a more rapid, machine-gun-like effect, shorten the delay loop in line 130.

### FASTER SID RESET

Listing 12 is a faster machine language program that resets all the SID registers except the Volume register. You should first run the program to poke the machine code into memory. Then, call the SID resetter with SYS 850 on a C-64 or with BANK 15: SYS 2816 on a C-128.

#### Listing 12. SID register reset program.

```

1Ø X=85Ø:REM FOR C64     :REM*182
2Ø IF PRE(Ø)<>(PRE(1) THEN X=28
16:BANK 15:REM FOR C128
                          :REM*183
3Ø FOR J=X TO X+13:READ A:POKE
J,A:NEXT                 :REM*25
4Ø DATA 162,23,169,Ø,157:REM*84
5Ø DATA Ø,212,2Ø2,2Ø8,25Ø
                          :REM*3Ø
6Ø DATA 141,Ø,212,96    :REM*46
    
```

Sound good? I hope so. And I hope you get a bang out of concocting computer sounds for your own programs and trying them on your friends! ■

*Bruce Jaeger has had scores of programs, articles and reviews published in many magazines, including RUN. He also plays bluegrass fiddle with the Middle Spunk Creek Boys in his native Minnesota.*

## NEW!

### A SUPERCONTROLLER MULTIFUNCTION CHIP!

THE



- DC DIMMER
- AC DIMMER
- 16 CHANNEL CONTROLLER
- 64 CHANNEL MATRIX
- PULSE COUNTER
- SERIAL IN / 16 CH OUT
- 4 CHASING ROUTINES
- SERIAL ENCODER
- SERIAL DECODER

In addition, the ZR2 provides:

3 OPERATION MODES—  
AUTO/SYNC/MANUAL;  
VARIABLE SPEEDS;  
100% SOFTWARE COMPATIBLE;  
REQUIRES ONLY +5V AND  
SINGLE CRYSTAL  
INTERFACES TO THE 64 & 128!  
THE ZR2 REPLACES MULTIPLE  
AND COSTLY DISCRETE PARTS.

IDEAL FOR ROBOTICS, LIGHT AND  
MOTOR CONTROL, MULTIMEDIA,  
DATA TRANSFER, AND MORE.

THE ZR2 IS TRULY A TOOL FOR  
THE TECHNICAL IMAGINATION!

ONLY \$34.95

NOW! NEW LOW PRICE!

THE ZR2 IS NOW \$29.95!



ALX DIGITAL 12265 S DIXIE HWY #922  
MIAMI FL 33156 305 553 3380

(please print clearly)

NAME \_\_\_\_\_

ADDRESS \_\_\_\_\_

CITY \_\_\_\_\_

STATE, ZIP \_\_\_\_\_

Send \$29.95 per ZR2 + \$1.55 Shipping; FLORIDA

RESIDENTS ADD 6% SALES TAX. Check or Money Order

Circle 412 on Reader Service card.

# It's All Relative

*If you're writing C-128 programs, you'll want the speed and ease of use relative files can offer.*



By ROB KENNEDY

**M**any owners of C-64s have probably tried using relative files, only to give up when they couldn't figure out the manual. Now, with the C-128, the problems are dissolved. Basic 7.0 includes many new commands that make relative files just as easy to use as sequential files, if not easier.

If you're new to computers or have never used files in programming, and you're wondering what relative files are, they are one of three main file types, the other two being sequential and program. A major distinguishing characteristic of both program and sequential files is that they are read from beginning to end. For example, let's say you had 100 addresses in a sequential file and you wanted to extract the information in address number 48. You'd have to read files 1-47 first, which could prove very slow. Relative files, on the other hand, can be read in any order, so they're a lot easier and faster to manipulate. If you want address 48, you just set the pointer to that location and get your information; you do not first have to read through files 1-47.

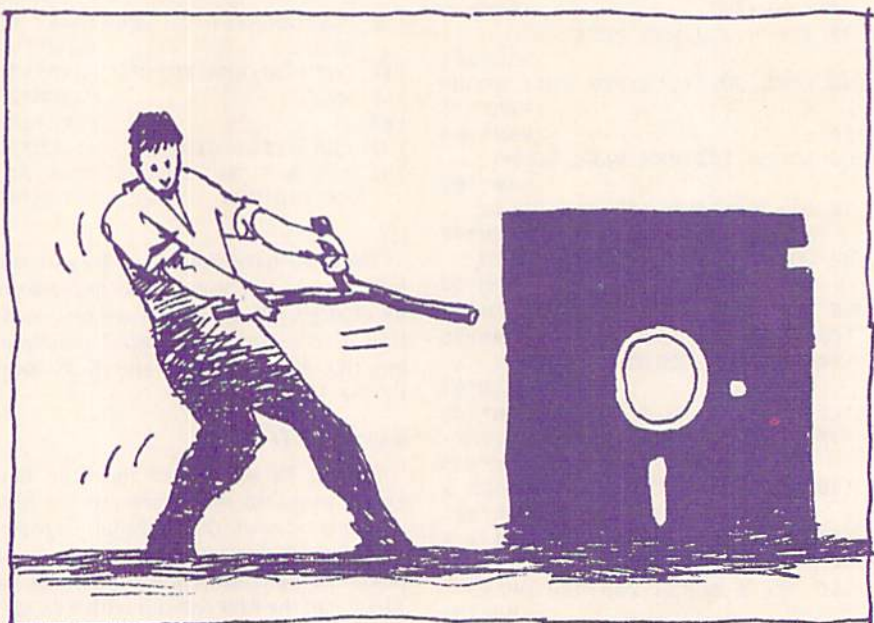
## CREATING RELATIVE FILES

Relative files are created on the C-128 with the DOpen command. Here's the format for the command, with brackets enclosing the descriptions of the parameter values you must supply:

```
DOPEN#[logical file number],"filename",
  L[record length],D[drive number],
  U[device number]
```

The logical file number ranges from 1 to 255. The record length is the number of characters you want in each record, with a maximum of 254. The drive number is 0 or 1 (this is optional; you only need to insert a value if you have a dual drive), and the device number (also optional) is usually 8.

To create a sample relative file that



we'll call REL FILE, put a formatted disk into the drive and type the following line:

```
DOPEN#1,"REL FILE",L30
```

Each record in REL FILE will be 30 characters long, but you don't have to use them all. The computer will just pad out the unused space with null characters. Note that whenever you're reading from or writing to a relative file, you must use the same logical file number as when you created it. Also, you should view the directory to note the length.

When writing to a relative file for the first time, you'll get a Record Not Present error, because the record pointer is set at a record that hasn't yet been written to. To prevent this problem, determine how many records you want to store, add one and then write something to the extra record. Let's say you want a file holding the names of the 50 states and their capitals; you'd need 100 records. That means you'd write a

"dummy" message at record 101.

Now type in DCLOSE on a separate line to close your file.

Next, type in the following program, noting that you don't have to specify the length of the file, because you did that when you created it:

```
5 REM ***PROGRAM ONE***
10 DOPEN#1,"REL FILE"
20 RECORD#1,101,1
30 PRINT#1,"FILE END"
40 PRINTDSS
50 DCLOSE
```

Now, run the program. Line 10 opens the file you created earlier, line 20 sets the record pointer (more on that later), and line 30 writes FILE END, the dummy message, to record number 101. Then line 40 displays the status of the disk error channel, and line 50 closes the file. It's always a good idea to display the disk error channel after reading from or writing to a relative file. Also, if you look at your directory, you'll notice ▶

# THERE ARE SOME THINGS YOU JUST CAN'T DO WITH GEOS... ...UNTIL NOW, THAT IS.

only  
\$24.97

Introducing **GEOS Power Pak\***, a collection of the most useful GEOS desktop accessories, utilities and applications ever assembled on one disk.

The editors of *RUN* magazine have packed this two-sided disk with over a half-dozen useful programs, a wide variety of fonts and over a hundred illustrations to use with GEOS. It features the work of some of the **BEST** talent in the GEOS market, including telecommunications expert Bill Coleman; font designers and artists Susan Lamb, Tom Trevor and Shaun Jones; and GEOS programmers Joe Buckley and Wayne Dempsey. This assures you, the GEOS user, of increased productivity and ease of use each and every time you boot up GEOS.

Discover how it feels to be a GEOS power user with the **GEOS Power Pak**. This disk will give you features unavailable anywhere else.

For example:

#### APPLICATIONS

—**geoTerm** is the first terminal program for GEOS. Before the **GEOS Power Pak**, this application had the experts stumped. But now you can telecommunicate to BBSs and online networks, sending and receiving messages, attending confer-

---

*"The editors of RUN have assembled the best talent in the GEOS community for this disk!"*

---

ences and uploading and downloading programs.

—**CardFile** is a file manager that comes in handy to maintain lists. Use it as an address book or to keep lists of tapes, records or household items... the possibilities are endless.

#### FONTS

—Choose from a wide selection of character fonts and point sizes. All the fonts are original and unpublished. Suitable for letter writing, headlines or to spruce up any newsletter, memo or sign.

#### CLIP ART

—Pick from over one hundred illustrations to use in your own documents.

#### ACCESSORIES

—**Thumbnail** is a unique program that reduces full-page geoPaint images for display on the screen and to save to disk. Catalog your geoPaint collection or paste images into a geoWrite document.

—**geoOrganizer** is a disk utility that lets you rearrange your GEOS files quickly and easily.

—**Pattern Editor** lets you create your own fill patterns for use within geoPaint.

—**geoBreak**. Enjoy this classic arcade game.

—**Convert 2.2**. Convert GEOS data and programs for uploading and downloading with geoTerm.

—**Write Hand Man**. Word and document analyzer.

—**AutoView**. geoPaint slide show maker.

—**PaintView II**. View geoPaint pages.

Like the original GEOS program, the **GEOS Power Pak** greatly enhances the capabilities of your C-64.

Let's face it. You've invested lots of time and energy learning to use GEOS. The **GEOS Power Pak** returns this investment tenfold with easy applications, expanded capabilities and increased productivity.

**GEOS Power Pak** revolutionizes the program that revolutionized your C-64.

\***GEOS Power Pak** is a product of *RUN* magazine and is not connected with Berkeley Softworks, creator of GEOS, or Commodore, manufacturer of the C-64.

**YES!** I want to add more computing power to my Commodore 64. Send me the **GEOS POWER PAK** from *RUN* magazine for \$24.97.

Check enclosed     American Express  
 MasterCard         Visa

Card # \_\_\_\_\_

Exp. Date \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_

State \_\_\_\_\_ Zip \_\_\_\_\_

Foreign Airmail, please add \$3.95 per order.

Mail to: **ReRUN 80 Elm St.**  
**Peterborough, NH 03458**

or call toll free **1-800-343-0728**

GPRS8

# PROGRAMMING

that the file is much longer than before. This is the maximum length.

## WRITING TO RELATIVE FILES

To write to a relative file, you have to open the file, set the record pointer and print to that record. Then, after you've printed the information, you should display the disk error channel and close the file.

Basic 7.0 provides several new commands for writing to relative files. One is the Record command, which sets the record pointer. Here's the format:

```
RECORD#[logical file number],[record number],[byte]
```

The logical file number is the same as the file number you opened with the DOpen command, which in turn is the same as the file number you used when creating the file. The record number designates the record in which you want to store information. If this were your first entry, you'd use record number 1; if it were the 25th state, you'd use record number 49 (remember, 24 states and capitals beforehand). The byte option tells where in the record you want to write (more about that later).

Now, let's write a little program for inputting the 50 states and capitals and then writing them into a relative file. Start with:

```
5 REM ***PROGRAM TWO***  
10 DOPEN#1,"REL FILE"
```

This line will open the file you created earlier. The longest state name is South Carolina, with 14 characters (including the space), and the longest capital name is Oklahoma City, also with 14 characters. Normally, a record length would probably be about 15, but you'll need an extra 15 characters later on, so we'll keep the record length at 30. Be sure to set a sufficient record length whenever you're creating a file, because any data that doesn't fit will be lost.

Now add the following lines to your growing program:

```
20 FORT = ITO100STEP2  
25 PN = PN + 1  
30 PRINT"STATE NUMBER";PN  
40 INPUT"WHAT IS THE NAME";ST$  
50 PRINT"CAPITAL NUMBER";PN  
60 INPUT"WHAT IS THE NAME";CA$
```

Line 20 starts a loop that will be executed 50 times, lines 30 and 50 display which state and capital are about to be entered, and lines 40 and 60 get the next state and capital. The state is stored in string variable ST\$, the capital in CA\$.

Continuing on, type in these four lines:

```
70 RECORD#1,T,1  
80 PRINT#1,ST$  
90 RECORD#1,T+1,1  
100 PRINT#1,CA$
```

The record pointer is set in lines 70 and 90, line 80 writes the state to that record and line 90 writes the capital to the next record. PRINT# is the command that does the actual writing. Here's its format:

```
PRINT#[file number],[print list]
```

The file number is the same as the file number opened by the DOpen command. The print list is what you want printed to the record, and it can be a string, a variable or a message in quotes. In our example, we're using a string. The next lines to enter are:

```
110 PRINTD$$  
120 NEXTT  
130 DCLOSE  
140 END
```

Line 110 prints the disk error channel, in keeping with the rule always to check the channel status. Line 120 is the end of the loop, and when it's been executed 50 times, the file will be closed by line 130.

When you write to a relative file, a section of its contents will look something like the first example in Figure 1. The same information stored in a sequential file would look like the second example in the figure. In both cases, I've used 0s to represent null characters and %s to represent boundaries between records.

These examples show why a relative file is faster. In a sequential file, the records do not all have the same length, so the computer has no way to find a particular record except by reading

through them all. With a relative file, every record is the same length, so the computer has reference points for finding particular ones.

## READING RELATIVE FILES

Now that you've written your information to a file, you want to get that information back. To do this, use almost the same program as for writing. The difference is an INPUT# command, which reads from a record instead of writing to it. Here's the format for the INPUT# command:

```
INPUT#[file number],[variable list]
```

The parameters are the same as for the PRINT# command.

Now, save your program, type in NEW and then enter the following lines:

```
5 REM ***PROGRAM THREE***  
10 DO  
20 INPUT"WHICH RECORD DO YOU WANT LOADED";RN  
30 IFRN<1ORRN>99ORINT(RN/2)*2 = RNTHEN20  
40 DOPEN#1,"REL FILE"  
50 RECORD#1,RN,1  
60 INPUT#1,ST$  
70 RECORD#1,RN+1,1  
80 INPUT#1,CA$  
90 PRINTD$$  
100 DCLOSE  
110 PRINT"THE CAPITAL OF "; ST$;" IS ";CA$;"."  
120 INPUT"MORE(Y/N)";AS  
130 IFA$<>"N"THENLOOP  
140 END
```

Line 10 sets up a loop, and line 20 asks which record you want. To make your selection, type an odd number from 1 to 99—odd because states are only in odd-numbered records. Line 30

Figure 1. Samples of file contents.

Sample 1. Relative file:

```
ALABAMA00000000%MONTGOMERY00000%ALASKA000000000  
Record# 1-----2-----3-----
```

Sample 2. Sequential file:

```
ALABAMA%MONTGOMERY%ALASKA  
Record# 1-----2-----3-----
```

Sample 3. Relative file with records divided into fields:

```
ALABAMA00000000MONTGOMERY00000%ALASKA000000000JUNEAU000000000  
Field# 1-----2-----1-----2-----  
Record# 1-----3-----
```



determines whether the value of RN is legal. If your typed value of RN is less than 1, greater than 99 or an even number, you'll be asked again.

Line 40 opens the file, lines 50 and 70 set the record pointers, and lines 60 and 80 get the information. Then line 90 displays the disk error channel, and line 100 closes the file. Finally, your information is displayed and you're asked if you want to get more. If so, execution loops back to the beginning; if not, the program ends.

## FILES, RECORDS AND FIELDS

So far, you've learned about files and records. Now to learn about fields. There are no special commands needed to manipulate fields; they're just other divisions within records, which you make yourself. Because your record length was set at 30 characters, you can store both a state and its capital in one record by making each field 15 characters long. When writing to your states file, add the capital to each state string, starting at character position 16 by us-

ing the Byte option in the Record command. The Record command lets you read from or write to a certain position in each record.

If you want to see this done, type in the following program, which will rewrite your file:

```
5 REM ***PROGRAM FOUR***
10 DOPEN#1,"REL FILE"
20 FORT = 1TO100STEP2
30 RECORD#1,T+1
40 INPUT#1,CA$
50 RECORD#1,T,16
60 PRINT#1,CA$
70 PRINTDS$
80 NEXTT
90 DCLOSE
100 END
```

Then execute the program and, when it's finished, load a copy of the third program above, which reads a file, and make this change:

```
70 RECORD#1,RN,16
```

Now line 70 sets the record pointer for the capital in the same record as

the state, but at a different character location—16. As a result, your relative file will look like the third example in Figure 1.

Notice that in this example I skipped over record #2. This is because, although the capital is still stored in that record, you aren't using it anymore. If you'd set up fields in the first place, there'd be no unused records.

To read this information, read the record normally. When the first string (the state) is found, the program will continue to line 70, where the record pointer is set to the second field. Line 80 will then get the capital.

This may seem like a lot of information to absorb in one sitting, but after you read over it a few times and experiment, you'll see how easy relative files actually are. You'll also find that, with simple modifications, the programs I've included here will take care of most of your relative file needs. ■

*Rob Kennedy is a freelance programmer pursuing a degree in computer science.*

## TYPE-IN TROUBLES?

YOU HAVE TYPED IN A *RUN* PROGRAM and are having some problems getting it to run. After a while, you feel like calling for help, but since we're not next door, it's expensive to call us. But we can share our experiences with you. Having heard from many users over the years about their difficulties with typing in listings, we've identified a few recurring problems that plague many people but are easy to fix. So read on and see if your problem is one of these. If so, perhaps the answers will help you find and correct the difficulty.

- You get an Out of Data in Line xxx message. This means that a program line was reading from Data statements and reached the end of the data before it was done reading. There are two possible problems.

- One might be with the line that reads the data, usually a For...Next loop. Make sure you have the proper values for the loop, because if the listing has a loop of 0 to 150 and you've typed 0 to 160, you'll get the "Out of Data" message. If the loop is correct, then the problem lies in the Data statements themselves. One possibility is that you omitted a whole line of data. That's easy enough to find and correct. More likely, you may have skipped one or more individual data items or typed in a period instead of a comma, which causes two data values to be read as one number. Check your typing carefully against the listing.

- You get an Illegal Quantity Error in Line xxx. That means that you've read a number from a Data state-

ment and tried to Poke it into a memory address. The error occurs because the number is larger than 255 (the largest value a memory address can contain), which means that somewhere in your Data statements you've made an error by typing in a number larger than 255. Again, this is easy to check for and correct. Just look in your Data statements for a number larger than 255. You might have added an extra digit, or perhaps you ran two numbers together (23456 instead of 234,56).

- You get a Syntax Error in Line xxx. This could be almost anything. What it tells you is that there is something wrong in the indicated line. Usually you've misspelled a Basic keyword or omitted some required character. List the line and examine it carefully.

- You get an Error in Data message. This occurs in programs that add up all the data as read, and, when finished, compares that sum with what it should be if all the data were typed in correctly. If it isn't the same, it means an error somewhere in typing the Data statements. Go back and check the data carefully, correct the mistake(s), save the new version and try again.

Finally, we urge everyone who intends to type in one of our listings to use *RUN's* Checksum program, which is printed in each issue. This nifty little program will help you avoid every mistake we mentioned above, except that it won't detect the omission of a line. ■

—LOU WALLACE

## PART 1

## Excuse the Interruption

*These advanced machine language programming techniques let you take full advantage of the excitement interrupts can provide.*



By JIM HOSEK

Every sixtieth of a second, your C-64 stops whatever it's doing and, with no awareness on your part, performs a series of important housekeeping functions, such as scanning the keyboard, blinking the cursor, and updating the software clock (TI and TIS). The computer does this through an interrupt, a process as close to true multitasking as the C-64 ever gets.

By playing some tricks in machine language, it is possible to intercept the interrupt and divert it to tasks other than housekeeping. In fact, this is how many games and machine language utilities accomplish sprite animation, background music, split graphics and text screens and the simultaneous display of more than eight sprites.

Part 1 of this article both explains the function and use of interrupts in the C-64 operating system and tells you how to generate and employ them in your own programs. Part 2 deals with the use and programming of the CIA chips to generate interrupts in your programs. Part 3 explores the use of interrupts originating from the VIC-II chip and from outside the C-64.

The accompanying examples and programs do require some understanding of 6510 machine language and an assembler or monitor program, but the listings are annotated to help beginners. (For more extensive aid in understanding machine language programming, consult *Machine Language for Beginners* and *The Second Book of Machine Language*, published by COMPUTE! Books. For general information on programming, see the *Commodore 64 Programmer's Reference Guide*, Commodore Business Machines, Inc.

#### THE COMPUTER'S SUBCONSCIOUS

Although a computer can't really think (not yet anyway), you might find

it useful to consider the C-64's 6510 microprocessor as analogous to the human brain, and the running of a program as the process of conscious thought. In such a comparison, then, you can think of an interrupt as a subconscious activity.

For example, when you're walking down the street, you don't consciously think about every step you take or about keeping your balance. More likely, you're thinking about a business problem, what to get your wife for her birthday or where you're going for lunch. If you thought about every step you took, the chances are that you'd fall flat on your face. It's impossible to think of two things at once, so your brain learned early on to leave the process of walking to some low-level area of your subconscious, so you don't have to "think" about it.

Interrupts work in a similar manner. The C-64's housekeeping routine, occurring 60 times a second, essentially diverts the 6510 microprocessor from whatever it's doing, preserves the current contents of the registers and accumulator, performs its tasks, and then lets the program resume operation.

#### INTERRUPT HARDWARE ANATOMY

It is the hardware component of the computer that generates interrupts, not the software, which involves the operating system and programs in ROM and RAM. The 6510 microprocessor has two sources of interrupts: the Interrupt Request (IRQ) line, and the Non-Maskable Interrupt (NMI) line. Both can receive interrupts from outside the C-64 through the expansion port. Internally, the two Complex Interface Adapter (CIA) chips and the Video Interface Controller (VIC-II) chip also generate interrupts. (See Figure 1).

The IRQ and NMI lines of the 6510

are normally high. That is, they have a +5V signal on them when they're inactive. In Figure 1, notice the squiggly lines with the words "Pull-up Resistor" next to them on both interrupt lines. These mini-circuits maintain the +5V signal. When any source drops to 0V, a voltage drop occurs across the pull-up resistor, and the voltage on the interrupt line also drops to 0V, generating an interrupt. This process is called an interrupt request.

IRQ interrupts come from the VIC-II chip, CIA 1 or pin 4 of the expansion port. CIA 1 is responsible for generating the IRQ request that occurs every sixtieth of a second in the 64's operating system by means of one of its built-in timers. NMI interrupts originate in the CIA 2, the restore key or pin D of the expansion port.

There are two parts to an interrupt. The first is the microprocessor or hardware part, occurring within the hardware programming of the 6510 CPU. The second portion is the software routine that determines the source of the interrupt and then performs whatever task has been requested.

Although interrupt requests may occur on either the IRQ or NMI lines, the 6510 handles them differently. Figure 2 shows a simplified flowchart of the sequence of events that occurs in the C-64 in response to an IRQ or NMI interrupt request.

#### THE IRQ REQUEST

Whichever line carries the request, the 6510 finishes the operation it's currently working on before acknowledging the interrupt. When it detects an IRQ interrupt request, the 6510 first checks the interrupt-disable status bit ("I" bit) of the processor status register (P). By setting this bit to 1, you can tell the microprocessor that you don't want ▶

RUN it right: C-64

Presenting:

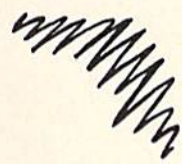
Special  
Disk  
offer  
5

# THE PERFECT COMPANION TO RUN'S SPECIAL PROGRAMMING ISSUE

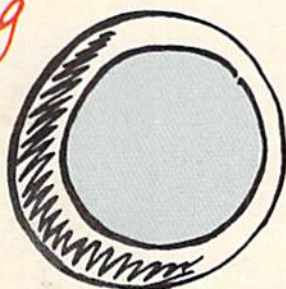
## THE DISK INCLUDES

- Gravitron
- Tag 'Em
- Astro Shoot
- C-64 Sprite Commands

3 arcade  
action!



Programming  
Tutorial



Handy  
Utilities

Fascinating  
Application

## PLUS, BONUS PROGRAMS

- Sprite Database
- C-128 Sprite Librarian
- A Show of Hands



Now, available only to RUN's Special Programming Issue readers, RUN is offering a collection of top-quality programs for the low, low price of \$7.95.

In all, three exciting arcade games, a sound utility for both the 64 and 128, some useful sprite utilities and a fascinating sign language tutorial.

Every major program in this issue is contained on this disk. For those of you who do not wish to type in program listings from this issue, this disk is a must. For those of you who are looking for some useful programs, this disk is a valuable addition to your software library. No hassle, no tedious hours of typing. Just load the disk and you're ready to go.

Send me \_\_\_\_\_ copies of the Special Programming Issue disk for the low price of \$7.95 for each disk ordered.

Check enclosed  MasterCard  American Express  VISA

Card # \_\_\_\_\_ Exp. Date \_\_\_\_\_ Signature \_\_\_\_\_

Name \_\_\_\_\_

Address \_\_\_\_\_

City \_\_\_\_\_ State \_\_\_\_\_ Zip \_\_\_\_\_

Foreign Airmail, please add \$3.95 per order.

Mail to: RUN Special Issue, 80 Elm St., Peterborough, NH 03458  
or call toll-free 1-800-343-0728

Yes!

## PROGRAMMING

any IRQ calls to occur now. The reasons you wouldn't want this to happen will be covered a little later. If you've set the bit to 1, the 6510 aborts the interrupt and executes the next instruction in the program.

You set the I flag by using the set interrupt-disable status (SEI) opcode

(\$78) and reset it with the clear interrupt-disable status (CLI) opcode (\$58).

If the I flag is 0, then the interrupt sequence begins. The processor status register (P) and program counter (PC) are pushed onto the stack. This will tell the 6510 where in the program it left off to deal with the interrupt. The in-

terrupt-disable status bit is set to 1 to prevent further IRQ requests from occurring.

Next, the program counter is loaded with the address stored at \$FFFE-\$FFFF in low-byte/high-byte format. In the C-64 operating system, this address is at the end of the Kernal ROM and points to address \$FF48. At this point, the hardware portion of the interrupt sequence is finished until control is returned to the interrupted program.

Now the software routine at \$FF48 takes over, first saving the contents of the .A, .X, and .Y registers by pushing them also on the stack. The routine next examines the break flag (B) of the P register. (Note: even though the contents of the P register were put on the stack, a copy still remains in the micro-processor.) If the B flag is set to 1, the interrupt routine knows it has encountered the BRK opcode (which uses the same pointer at \$FFFE-\$FFFF), and it hands control over to the BRK routine at \$FE66.

If the B flag is 0, then the rest of the IRQ routine is executed, with an indirect JMP through a vector in RAM at locations \$0314-\$0315. This vector points to the necessary housekeeping routine (at \$EA31), which is responsible for the cursor-blinking and keyboard-reading activities. Because this vector is in RAM, you can use it to divert control to your own IRQ routine. I'll discuss this technique shortly.

Before the interrupted program regains control, the .A, .X, and .Y registers are retrieved, unchanged, from the stack. The return-from-interrupt (RTI) opcode signals the end of the interrupt routine and sends control back to the 6510 routine. Like the return-from-subroutine (RTS) instruction, the PC is loaded with the return address that was saved on the stack. Unlike an RTS, however, the P register is also pulled off the stack.

In essence, the microprocessor is completely restored to the state it was in before the interrupt. The only evidence the program has that something happened is that a few memory locations may have changed and the keyboard buffer might contain another jump. The IRQ request is almost like a Jump-to-Subroutine (JSR) command, except that an IRQ is called by hardware, not software, it always goes to the same place, and the contents of the P register are also preserved.

### THE NMI REQUEST

An NMI request is handled slightly differently. As its name (Non-Maskable

Figure 1. Interrupt lines and sources in relation to the 6510 CPU.

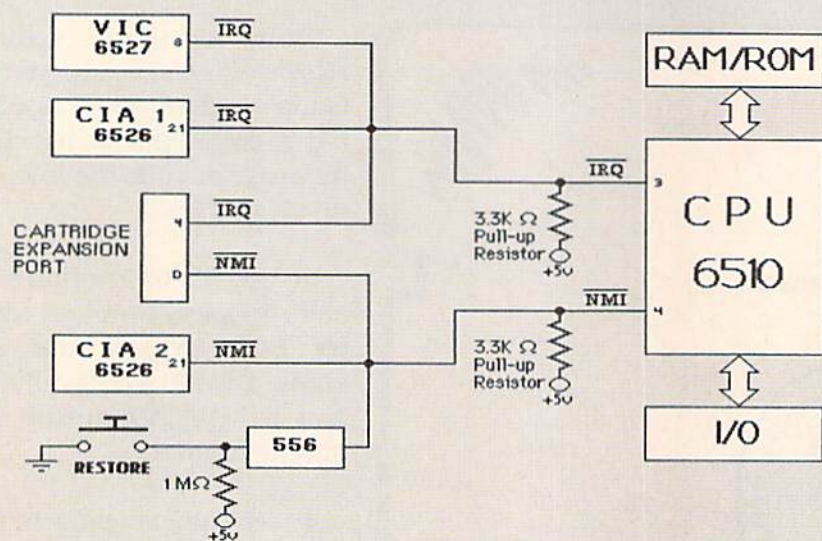
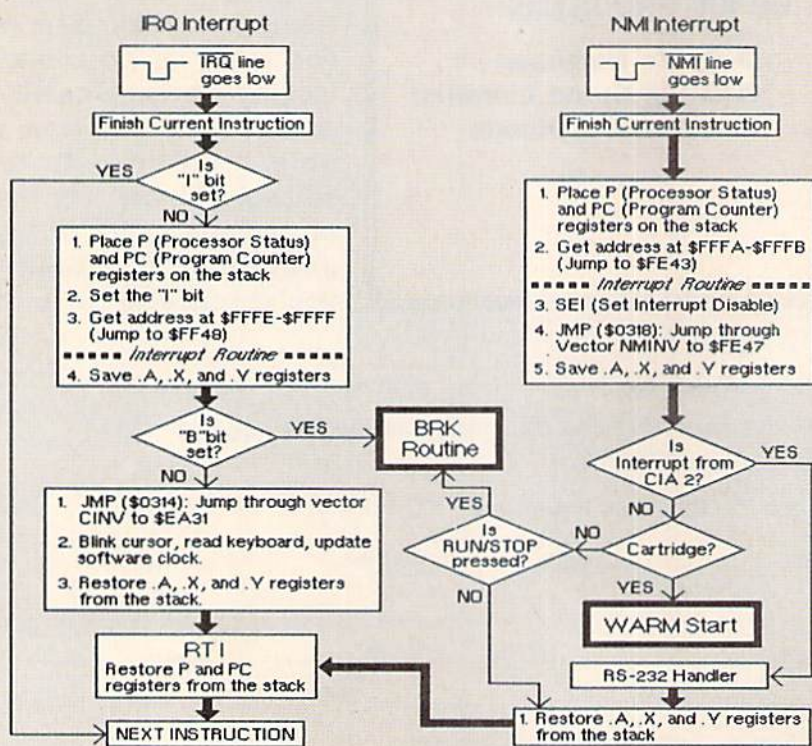


Figure 2. Flowchart of sequences triggered by IRQ or NMI interrupt requests.



# PROGRAMMING

Interrupt) implies, an NMI request may occur whether or not the I flag is set. You might compare this to putting a phone call on hold in order to answer the other line. It is possible for an NMI to occur while an IRQ or even another NMI is being executed.

As with the IRQ request, the 6510 pushes the P and PC registers onto the stack, but, unlike the IRQ, it does not test or set the I flag. The CPU then looks to addresses \$FFFA-\$FFFB for the location of the NMI routine, which starts at \$FF43. Here, also unlike the IRQ routine, it uses the SEI opcode to set the I flag to prevent IRQs from occurring, and the program jumps through a RAM vector at \$0318-\$0319, pointing to \$FF47, before saving the .A, .X, and .Y registers. This fact is important to consider when you start diverting the NMI routine for your own purposes.

Next, the interrupt routine tries to discover the source of the interrupt. It looks at the interrupt control register (ICR) of CIA 2. If it finds the seventh bit set, it diverts control to the RS-232 input/output routine. Here, it checks the RS-232 interface to see if it's ready to send or receive more data. The NMI routine then ends like the IRQ, with the .A, .X, and .Y registers restored, and an RTI follows.

If CIA 2 was not the source of the interrupt, the routine assumes that it was the restore key. It checks locations \$8004-\$8008 for the Autostart ROM Cartridge code (CBM80), and if a cartridge is present, the routine is exited through the cartridge warm start vector at \$8002-\$8003.

If no cartridge is present, the routine checks the run-stop key, and, if that key has been pressed, it jumps to the BRK routine used by the BRK opcode. So pressing the run-stop/restore combination has the same effect as the 6510 encountering a BRK opcode.

If the run-stop key has not been pressed, the program jumps to the end of the routine, where the microprocessor registers are restored, and then to the RTI instruction, which, of course, returns control to the interrupted program.

## OTHER INTERRUPT USES

The C-64 operating system also uses the IRQ interrupt for cassette tape operations by means of CIA 1. In essence, a tape Save or Load command will send the IRQ routine to another routine by means of the RAM vector discussed earlier. The CIA chip Timer A is reprogrammed from its normal sixtieth-of-a-second interrupt to work for the tape routine. After the operation is com-

plete, the RAM vector and CIA chip revert to their former values.

The IRQ interrupt is also used during the power-up sequence to determine whether the American NTSC or the Eu-

ropean PAL video system is in use. This is important in determining the setting of the sixtieth-of-a-second interrupt timer, since each system uses a different clock speed. The check is done by pro-

### Listing 1. Character-cycling IRQ program.

```

SEI                ; Set interrupt Disable Flag
LDA #<NEWIRQ      ; Get low-byte of NEWIRQ routine
STA $0314         ; Store in RAM vector
LDA #>NEWIRQ      ; Get low-byte of NEWIRQ routine
STA $0315         ; Store in RAM vector
CLI               ; Clear Interrupt Disable Flag
RTS               ; Done
NEWIRQ
INC $05A4         ; Increment character at center of
                  ; screen
JMP $EA31        ; Jump to regular IRQ routine
    
```

### Listing 2. Color-changing NMI program.

```

LDA #<NEWNMI      ; Set up for New MNI
STA $0318
LDA #>NEWNMI
STA $0319
RTS
NEWNMI
INC $D020         ; Increment Border Color register
INC $D021         ; Increment Background Color register
JMP $FE47 (FA40) ; Jump to regular NMI routine
    
```

128  
w  
64

### Listing 3. Ball-Animation program.

```

*= $C000          ; Start address
                  ; Ball Animation
                  ; Interrupt demo
                  ; Set Interrupt Disable Flag
                  ; Change IRQ RAM vector
SEI
LDA #<NEWIRQ
STA $0314
LDA #>NEWIRQ
STA $0315
CLI               ; Clear Interrupt Disable Flag
LDX #$10         ; Set Sprite Location Registers
ILOOP
LDA SP,X         ; ($D000-$D010) set to initial values
STA $D000,X     ; From table SP
DEX
BPL ILOOP
LDX #$07         ; Set Sprite Color Registers
CLOOP
LDA SC,X         ; ($D027-$D02E) to values in table SC
STA $D027,X
LDA #$0B         ; and Sprite Pointers ($07F8-$07FF)
STA $07F8,X     ; to sprite block 11
DEX
BPL CLOOP
LDA #0           ; Initialize other VIC registers
STA $D017       ; X-expand
STA $D01C       ; Multicolor
STA $D01D       ; Y-expand
STA $D020       ; Border Color (Black)
STA $D021       ; Background Color (Black)
LDA #$FF
STA $D015       ; Sprite Enable Register
LDX #$3F       ; Put Sprite Data in Block 11
                  ; ($02C0-$02FF)
SLOOP
LDA BALL,X      ; Data in Table BALL
STA $02C0,X
DEX
BPL SLOOP
JSR EXP         ; (For further expansion)
RTS
NEWIRQ
NOP             ; (For further expansion)
NOP
NOP
    
```

# PROGRAMMING

```

CONT
  LDX #$0F          ; Set X to $0F - end of Sprite
                   ; Position
MLOOP
  TXA              ; Main loop - save bit 0
                   ; (0=X value, 1=Y value)
  LSR              ; Put in Carry Flag
  PHP              ; Save Carry Flag
  TAY              ; Set Mask Bit for MSB register
($D010)
  SEC
  LDA #0
BLOOP
  ROL
  DEY
  BPL BLOOP
  TAY              ; Store Mask in Y register
  PLP              ; Retrieve Carry Flag (Bit 0)
  LDA SD,X         ; Get direction of corresponding
                   ; position register
  BPL REGINC       ; If increasing, go to REGINC
  DEC $D000,X     ; Decrease value of position register
  BNE NEXT         ; If > 0 then do next register
  BCS NEXT         ; If a Y-position, then do next
                   ; register
  TYA              ; Get MSB register mask
  EOR $D010        ; Toggle appropriate bit of MSB
register
  STA $D010
  TYA              ; Get MSB mask again
  AND $D010        ; Test appropriate bit of MSB register
  BEQ NEXT         ; If not set then do next register
  LDA #$5B         ; Load A with $5B (91), Put in X
position
  STA $D000,X     ; Register if sprite wraps around
                   ; left side
  JMP NEXT         ; Next register
REGINC
  INC $D000,X     ; Increase value of position register
  BCS NEXT         ; If a Y position, do next register
  BNE CHECKX      ; If not 0 check right edge of screen
  TYA              ; Get mask bit
  ORA $D010       ; Set MSB
  STA $D010
  JMP NEXT         ; Do next register
CHECKX
  LDA $D000,X     ; Get current X position
  CMP #$5B        ; Compare to $5B (91)
  BNE NEXT         ; If not 91 then do next register
  TYA              ; Get mask
  AND $D010       ; Test MSB register
  BEQ NEXT         ; If not at right edge, do
                   ; next register
  LDA #0          ; Put sprite at left edge
  STA $D000,X
  TYA
  EOR $D010
  STA $D010
NEXT
  DEX              ; Decrement X - point to next position
                   ; register
  BPL MLOOP       ; If not done, do again
  JMP $EA31       ; Go to KERNAL IRQ routine
SP
  .BYTE 155 90 205 95 255 130 250 175      ; Initial
                                           ; positions
SC
  .BYTE 205 205 155 210 110 170 105 135 0
SD
  .BYTE 8 9 10 11 12 13 14 15             ; Sprite colors
BALL
  .BYTE 0 128 128 0 128 128 128 0        ; Direction
                                           ; 0=INC, 128=DEC
  .BYTE 0 0 128 128 0 128 0 0
  .BYTE 0 0 0 1 255 128 7 31             ; Sprite Data
  .BYTE 224 12 255 240 31 255 248 63
  .BYTE 15 252 126 15 254 124 15 254
  .BYTE 248 112 63 248 240 255 248 255
  .BYTE 255 248 240 255 248 112 63 124
  .BYTE 15 254 126 15 254 63 15 252
  .BYTE 31 255 248 15 255 48 7 248
  .BYTE 224 1 255 128 0 0 0 0
EXP
  RTS

```

programming the VIC-II chip to generate a raster-compare IRQ if the raster screen line reaches 311, more lines than the NTSC system possesses. If a raster interrupt occurs, the timer is adjusted to the PAL standard. (I'll discuss raster-compare interrupts in greater depth in Part 3.)

As mentioned before, the CIA and VIC-II chips are the C-64's main sources of interrupts. You can program them to generate interrupt requests for many conditions, including an alarm function built into the CIA chips, the presence of an outside signal, or the switching of an optical transistor in the tip of a light pen connected to the VIC-II chip.

## PROGRAMMING INTERRUPTS

The easiest way to get your own interrupt routine up and running is by diverting to your own routine the RAM vectors at \$0314-\$0315 for the IRQ interrupt and at \$0318-\$0319 for the NMI.

Listing 1 is a short program that uses the IRQ interrupt that the CIA chip generates. The first instruction is the SEI opcode. This, of course, sets the I flag of the P register to 1 and prevents any IRQ interrupts from occurring. Since you're changing the RAM vector, an IRQ interrupt halfway through might make the IRQ routine jump into limbo and crash.

Next, you load the address of the new IRQ routine into locations \$0314-\$0315 in low-byte, high-byte format. When this is finished, the I flag is cleared, and the set-up portion of the program is complete. From now on, the RAM vector will direct the IRQ routine to your routine at \$C00D every sixtieth of a second.

The routine NEWIRQ simply increments location \$05A4 (which is in the middle of the text screen), then jumps to the normal IRQ routine at \$EA31.

Type in this program and compile it if you're using an assembler. If you're using a machine language monitor, just enter the opcodes, ignoring the labels. Then enter SYS 49152. You should see a space in the center of the screen cycling through all 256 characters of the character set about once every four-and-a-quarter seconds.

Try loading or saving a program. The speed at which the characters change during these operations will become slow and irregular. This is because the serial bus routines that perform these operations will occasionally set the interrupt disable during critical timing sequences. Pressing the run-stop/restore combination resets the RAM vec-

## PROGRAMMING

tor and stops the changing characters in the middle of the screen.

The program in Listing 2 is similar to that in Listing 1, except it intercepts the NMI-routine RAM vector. Note that you don't use the SEI and CLI instructions, simply because they have no effect on NMIs. There is some small risk that the computer will crash while altering this vector, but if no RS-232 device is present and you stay away from the restore key, there should be no problem. The second program sits right above the first.

The NEWNMI routine increments the border and background color registers of the VIC-II chip every time an NMI interrupt occurs; then it jumps to the normal NMI routine. Type in and compile the program, and then activate it by typing SYS 49171. Nothing should happen as yet. Now tap on the restore

key a few times. (Do *not* hold down the run-stop key.) The screen will change colors with every tap.

Now try to get both new interrupt routines running by typing SYS 49152 again. Press the restore key a few times to convince yourself that both new routines are installed.

This is basically how you can utilize interrupts on the C-64. It is also possible to put your own vectors in memory at \$FFFA-\$FFFB and \$FFFE-\$FFFF by switching off the Kernal ROM and writing your own routines to handle the entire interrupt. But you'll probably find it much easier to have the Kernal ROM around to handle some of the customary chores of the interrupt.

### NOW WHAT?

By now, you should be getting some ideas about ways to use interrupts in

your own programs. Listing 3 is a short program that animates eight colored balls on the screen and has them flying all over. It runs on the IRQ interrupt, utilizing the sixtieth-of-a-second timer generated by CIA 1. Look through the listing carefully and try to understand what is being done and why. The setup portion is similar to program 1, but the NEWIRQ routine constantly affects the sprite location registers, thus creating the animation effect.

You activate the program by typing SYS 49152. Note that even while the balls are flying around, you can still type at the keyboard and even run or list other programs. The animation is done entirely in the background. You can also use this technique to run two programs at the same time, creating a multitasking effect. Such is the power of interrupts. ☐

## PART 2

### THE CIA CHIP

There are two CIA chips in the C-64: one wired to the 6510 microprocessor's IRQ line and the other to the NMI line. As mentioned in Part 1, interrupt requests can be generated through this mysterious, yet powerful, chip.

The CIA chip is like a miniature computer that runs alongside the 6510 CPU. Think of the CIA—or 6526, as Commodore calls it—as a peripheral processing unit (PPU). It doesn't depend on the 6510 for any of its functions, but helps support it, as well as the 64's operating system.

Like the 6510, the CIA chip can be programmed to a limited extent to carry out specific tasks. Its primary function, however, is to handle input/output tasks and thus help free up the 6510.

Unlike the 6510, the CIA cannot read or write directly from or to RAM and cannot read from ROM. Instead, it's programmed by reading or writing to its 16 internal registers. Table 1 gives a brief description of these registers, which, in the C-64, are accessed by means of memory locations \$DC00-\$DC0F (56320-56335) for CIA 1 and \$DD00-\$DD0F (56576-56591) for CIA 2.

The CIA chip uses the registers to communicate with the 64. Through the IRQ line, it can also tell the 6510 when one of its five sources of interrupts has been activated.

### INTERRUPT SOURCES

Figure 3 shows the sources of interrupts from a CIA chip. Timer A on CIA

1 is set to give an interrupt every sixtieth of a second for the operating system's housekeeping routine. Timer B of CIA 1 is for serial bus timing. Timers A and B of CIA 2 are used for RS-232 I/O.

One interrupt source, the time-of-day (TOD) clock, isn't used by the C-64's operating system. Since the clock is part of the hardware, it's far more accurate than the software clock (TI and TIS) maintained by the sixtieth-of-a-second housekeeping routine, and it's not subject to disruption from disk operations. Because it's accurate to one-tenth of a second, in contrast to the software clock's one second, the TOD clock is used for applications that require critical timing.

The serial port, also unused by the C-64's operating system, is another source of interrupts. It generates an interrupt request after it has received eight bits of serial data or has finished sending eight bits. Then, it's up to the interrupt routine to read the data from the port's shift-register or load the register with the next eight bits to be sent.

The serial port is accessed at the user port through lines SP (CIA 1, pin 5; CIA 2, pin 7) and CNT (CIA 1, pin 4; CIA 2, pin 6).

The most powerful sources of interrupts are timers A and B. Each timer is a 16-bit counter, capable of counting microprocessor clock cycles or external pulses on line CNT of the user port. In addition, timer B can count the number of times that timer A goes to zero. This feature allows the timers to be used separately to generate interrupts of

short intervals, or together for interrupts with intervals of up to 70 minutes.

The timers can also operate in Continuous or One-Shot mode. Either way, an interrupt can be generated every time the counter goes to zero.

In Continuous mode, the timer resets itself to the value stored in the timer latch register. Here, the timer can be compared to an electric clock's snooze alarm that wakes you up every five minutes. In One-Shot mode, the timer counts to zero and then stops.

The flag line can generate interrupts from outside the C-64. Because it doesn't connect directly to the IRQ or NMI line on the 6510, the flag is advantageous in that the interrupt routine can determine the source of the interrupt, and even turn off its ability to generate one.

CIA 1 has its flag line connected to SRQ IN for serial bus operations. CIA 2's flag line isn't used by the 64, but is available at pin B of the user port.

### GETTING INTERRUPTS TO THE 6510

All five sources of interrupts can be detected by means of the interrupt control register (ICR \$0D) of the CIA chip. For CIA 1, this register is located at \$DC0D (56333), while CIA 2's is located at \$DD0D (56589). The ICR also provides a means of determining which source's interrupts will actually generate an interrupt request on either the IRQ or NMI line.

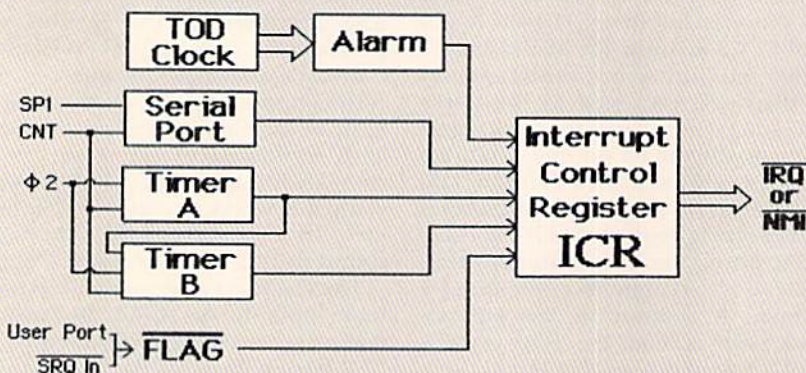
Figure 4 illustrates how the ICR is set up. Its function depends on whether you're reading or writing to it. ►

# PROGRAMMING

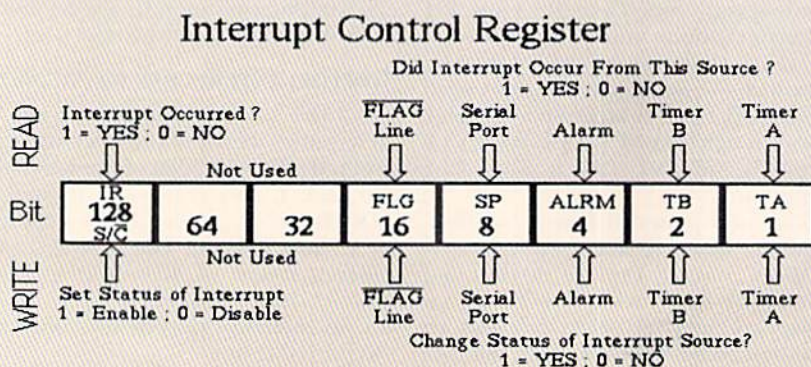
**Table 1. Complex Interface Adapter Register Map**

REGISTER	ADDRESS	DESCRIPTION
PRA	\$00	Port A: 8-bit bidirectional Peripheral Data Register
PRB	\$01	Port B: 8-bit bidirectional Peripheral Data Register
DDRA	\$02	Port A: Data Direction Register
DDRB	\$03	Port B: Data Direction Register
TA LO	\$04	Timer A: low byte
TA HI	\$05	Timer A: high byte
TB LO	\$06	Timer B: low byte
TB HI	\$07	Timer B: high byte
TOD 10THS	\$08	TOD clock: tenth of a second
TOD SEC	\$09	TOD clock: seconds
TOD MIN	\$0A	TOD clock: minutes
TOD HR	\$0B	TOD clock: hours
SDR	\$0C	Serial Data Port
ICR	\$0D	Interrupt Control Register
CRA	\$0E	Control Register A
CRB	\$0F	Control Register B

**Figure 3. Sources of interrupts from a CIA chip.**



**Figure 4. The interrupt control register.**



When you're writing to it, you can enable or disable the interrupts from each of the five potential sources, depending on the setting of bit 7. You can program any source to generate interrupts, but the microprocessor will never see the interrupt request unless the appropriate bit of the ICR is set or enabled.

When bit 7 equals 1, then an interrupt is enabled or set if its corresponding bit is set to 1. For example, writing \$81 (129, %10000001) to the ICR enables interrupts from timer A. The status of the other four sources remains unchanged. In order to disable or clear an interrupt, bit 7 must be set to 0, and the corresponding source's bit set to 1. Writing \$1F (31, %00011111) disables all five sources of interrupts. The values of bits 5 and 6 have no effect during either operation.

When reading the ICR, the status of all the interrupt sources can be determined. Bit 7 is used to signal if an interrupt has occurred from one of the possible sources on the CIA chip.

The C-64 operating system checks bit 7 during the NMI interrupt routine to see if the interrupt was generated by the RS-232 interface (7 equals 1) or the restore key (bit 7 equals 0). Because the restore key is connected directly to the NMI line of the 6510 microprocessor, its interrupt bypasses CIA 2 and has no effect on the ICR.

Furthermore, the bit corresponding to the source of the interrupt from the CIA chip is also set. Therefore, if the interrupt from the TOD clock alarm is enabled, and the time in the alarm register matches it, an interrupt request is sent to the 6510 and the ICR register is set to \$88 (136, %10001000). Note that when the ICR is read, its contents are set to zero, so if there's more than one source of an interrupt to check, a copy of the register should be saved to RAM.

## GENERATING INTERRUPTS

I'll now proceed to detail the methods of generating interrupts from the various sources mentioned above.

**Time-of-day clock alarm**—Getting the TOD clock alarm to generate an interrupt involves setting up an interrupt routine, changing the appropriate RAM vector and programming the CIA chip.

The TOD clock occupies registers \$08-\$0B of the CIA chip. In the 64, these are locations \$DC08-\$DC0B (56328-56331) for CIA 1, and \$DD08-\$DD0B (56584-56587) for CIA 2. These registers handle tenths of a second, seconds, minutes and hours. The hours register also contains an AM/PM flag.



## PROGRAMMING

Figure 5 shows how these registers are arranged. The values are stored as binary-coded decimals (BCD). This means that the first four and last four bits each represent one digit, from zero to nine. Therefore, a single byte can contain a value from zero to 99 instead of zero to 255, as is the case with ordinary binary. The decrease in range is compensated for by facility in reading and displaying program values.

With some of the registers, only part of a nibble (half a byte, or four bits) is used. In these cases, all necessary values for that digit can be represented by the space allowed.

Reading the registers returns the current time. The TOD clock has a built-in latching feature that freezes the time whenever the hours register is read. This prevents the time from "rolling over" after only part of it has been read. An analogy to this is the lap feature found on electronic stopwatches. When you press the lap button, the time display freezes so you can tell the speed of a runner during part of a race. Meanwhile, the stopwatch is keeping time internally. Pressing the button again shows elapsed time.

The TOD clock registers resume exhibiting the correct time after the tenths-of-a-second register is read. It's important to read this register last and the hours register first.

Setting the current and alarm time is done by writing to the registers. Once again, a latching function is in effect. The clock freezes when the hours register is written to, and starts with the new time when the tenths-of-a-second register is written to. Bit 7 of register \$0F (\$DC0F or \$DD0F for CIA 1 or 2) tells the CIA chip if you're entering the current time or the alarm time. If the bit is set to 0, then the clock is set to the specified time; if it's set to 1, then the alarm time is set.

Listings 4 and 5 take care of all these tasks when setting up the CIA 1 alarm using the IRQ interrupt. Listing 4 is in machine language. It is responsible for changing the RAM vector at \$0314-\$0315, and it also contains the new interrupt routine.

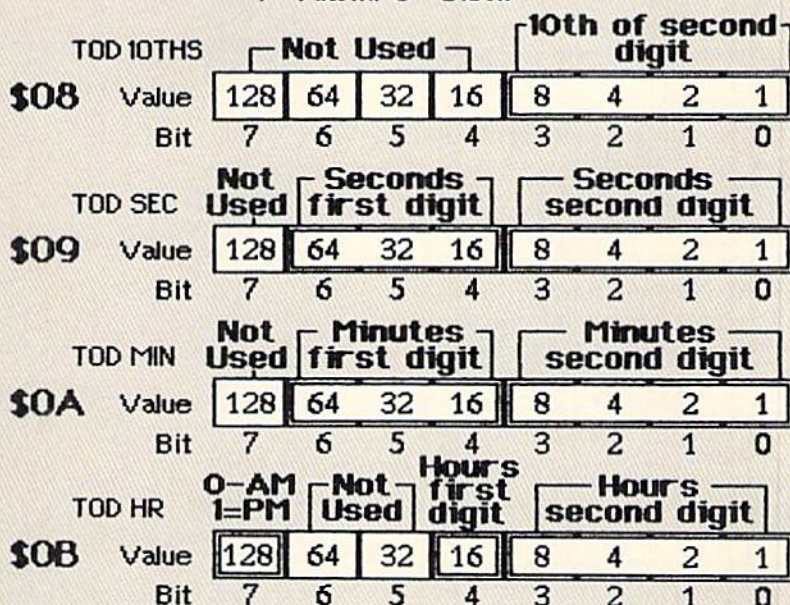
The new routine first updates the time in the upper-right corner of the screen from the TOD clock registers (lines 1150-1680). It then checks the ICR of CIA 1 (\$DC0D) for bits 7 and 2:

```
1660 LDA $DC0D
1670 AND #$84
1680 CMP #$84
1690 BEQ ALARM
1700 JMP $EA31
```

Figure 5. The time of day clock and alarm.

### Time Of Day Clock and Alarm

Read: Returns Current Time  
Write: Sets Clock time or Alarm time  
Depending on setting of Bit 7 of CRB  
1 = Alarm; 0 = Clock



Listing 4. TOD clock alarm demo.

```
*=$C000
; ****ALARM CLOCK****
; TOD CLOCK ALARM DEMO
;
SEI ; Set Interrupt Disable Flag
LDA #<NEWIRQ ; Change RAM IRQ Vector
STA $0314
LDA #>NEWIRQ
STA $0315
CLI ; Clear Interrupt Disable Flag
RTS
NEWIRQ
BIT $DC0B ; Test AM/PM flag
BMI PM ; If not PM...
LDA #$81 ; Get a reverse "A"
BNE APFLAG ; Jump ahead
PM
LDA #$90 ; Get a reverse "P"
APFLAG
STA $0427 ; Put character in upper left corner
LDA $DC0B ; Get first digit of HOURS
AND #$10 ;
BEQ HOURO ; Check if it is "0"
LDA #$B1 ; Get a reverse "1"
BNE D1H ; Jump ahead
HOURO
D1H
LDA #$B0 ; Get a reverse "0"
STA $041D ; Put up first digit on screen
LDA $DC0B ; Get second digit of HOURS
AND #$0F ; Mask out bits 4-7
ORA #$B0 ; Add on #$B0 to get reverse number
STA $041E ; Put up second digit
LDA #$BA ; Get a reverse ":"
STA $041F ; Put on screen
LDA $DC0A ; Get first digit of MINUTES
AND #$70 ; Save bits 4-6
```

# PROGRAMMING

If the interrupt is not from the TOD clock alarm, then control is passed to the operating system's housekeeping routine. See how simple the BCD format of the TOD clock makes it to display the time?

If the interrupt is from the alarm, the routine uses the SID chip to generate an alarm sound. It flashes different colors on the screen and returns control to the program that was interrupted. Since the normal interrupt routine is not accessed, the registers must be restored from the stack before the RTI instruction is executed.

Listing 5 is in Basic. Using the filename "ALARM.IRQ", it loads Listing 4 into memory. It is also used to set the TOD clock and the alarm, and to enable the alarm interrupt. Once the necessary registers have been set, the program is no longer needed. Pressing the run-stop/restore resets the RAM vector and CIA chip and turns off the alarm, so *don't* hit this combination before the alarm sounds.

The program will ask you if you wish to set the clock time, or the alarm time or whether you want to quit. When entering the time, only the number keys, A, P, cursor right and return are active. Once the time appears as you want it set, press return.

Try setting the time and alarm several minutes apart to see how it works. Make sure you understand what every part of Listings 4 and 5 are doing and which of the CIA registers these programs are affecting.

Incidentally, GEOS owners, this is essentially how deskTop's clock and alarm work. GEOS also uses CIA 1's TOD clock.

**Timers A and B**—Programming the CIA chips' timers is a little more complicated. I'm limiting my discussion to their use as sources of interrupts while they're set to count internal microprocessor clock cycles. As previously mentioned, the timers can also count pulses from an external source on line CNT of the user port.

The two 16-bit counters that make up timers A and B are located in registers \$04-\$05 and \$06-\$07, respectively, in low-byte/high-byte format. These are addresses \$DC04-\$DC07 (56324-56327) for CIA 1, and \$DD04-\$DD07 (56580-56583) for CIA 2. Reading these registers returns the current value of the appropriate counter; writing to them loads the timer latch.

When started, the counters count down once every clock cycle. On the C-64, there are 1,022,730 cycles per second, based on the NTSC system. You

```

LSR                               ; Shift right 4 bits
LSR
LSR
LSR
ORA #$B0                           ; Get reverse character
STA $0420
LDA $DC0A                           ; get second digit of MINUTES
AND #$0F
ORA #$B0
STA $0421
LDA #$BA
STA $0422
LDA $DC09                           ; Do the same for SECONDS
AND #$70
LSR
LSR
LSR
LSR
ORA #$B0
STA $0423
LDA $DC09
AND #$0F
ORA #$B0
STA $0424
LDA #$AE                           ; Get a reverse "."
STA $0425                           ; Put on the screen
LDA $DC08                           ; Get Tenth of seconds
AND #$0F
ORA #$B0
STA $0426                           ; Put on screen
LDA $DC0D                           ; Check ICR if interrupt
                                       ; is from the Alarm
                                       ; Check bits 7 and 2
AND #$84
CMP #$84
BEQ ALARM                           ; If both are set, sound alarm
JMP $EA31                           ; Go to KERNAL IRQ routine
ALARM
LDX #$10                           ; Setup for Alarm (16 loops)
LDY #$00                           ; Set Delay Counters
STY $02
LDA #$20                           ; Set up SID chip
STA $D400                           ; FREQ Lo
STA $D401                           ; Freq Hi
LDA #$0F
STA $D418                           ; Volume
LDA #$00
STA $D406                           ; S/R
LDA #$29
STA $D405                           ; A/D
ALOOP1
LDA #$11                           ; Main Alarm LOOP
STA $D404                           ; Turn on Triangle Waveform
ALOOP2
DEY                               ; Delay LOOP
BNE ALOOP2
DEC $02
BNE ALOOP2
DEC $D020                           ; Decrement border color
DEC $D021                           ; Decrement background color
LDA #$10
STA $D404                           ; Turn off voice 1
DEX
BNE ALOOP1                           ; Decrement LOOP counter
                                       ; If not done, do it again
PLA                               ; Restore registers from the stack
TAY
PLA
TAX
PLA
RTI                               ; Return to program

```

**Listing 5. Basic program that activates Listing 4.**

```

100 REM - TOD CLOCK ALARM DEMO           :REM*206
110 REM                                 :REM*253
120 L(1)=1:L(2)=9:L(4)=5:L(5)=9:L(7)=5:L(8)=9:L(10)=9 :REM*219
130 POKE53280,0:POKE53281,0:PRINT"(SHFT CLR){CTRL 8}"; :REM*115
140 IFA=0THENPRINT"LOADING ALARM.IRQ":A=1:LOAD"ALARM.IRQ",8,1 :REM*178
                                       :REM*178
150 SYS 49152                           :REM*69
160 PRINT"{SHFT CLR}{3 CRSR DNs}{2 CRSR RTs}{CTRL 8}1. SET CLOC :REM*81
    K TIME"

```

# PROGRAMMING

```

170 PRINT"{2 CRSR RTs}{CRSR DN}2. SET ALARM TIME" :REM*143
180 PRINT"{2 CRSR RTs}{CRSR DN}3. QUIT" :REM*8
190 GETB$:IFB$<"1"ORB$>"3"THEN190 :REM*53
200 IFB$="1"THENCB=0:T$="CLOCK" :REM*49
210 IFB$="2"THENCB=128:T$="ALARM" :REM*48
220 IFB$="3"THENEND :REM*176
230 PRINT"{CTRL 6}PLEASE ENTER THE "T$" TIME":H=1:AP$="A":M=0:S
=0 :REM*216
240 PRINT"{10 CRSR RTs}01:00:00.0A{11 CRSR LFs}";:TM$="01:00:00
.0A":CP=1 :REM*187
250 PRINT"{CTRL 9}"MID$(TM$,CP,1)"{CTRL 0}{CRSR LF}"; :REM*44
260 GETA$:IFA$=" "THEN260 :REM*178
270 IFA$=CHR$(13)THEN380 :REM*85
280 IFA$="{CRSR RT}"THEN340 :REM*168
290 IFCP<11THEN320 :REM*196
300 IFA$<"A"ANDAS<"P"THEN260 :REM*119
310 GOSUB330:GOTO260 :REM*148
320 IFA$<"0"ORAS>CHR$(L(CP)+48)THEN260 :REM*135
330 TM$=LEFT$(TM$,CP-1)+A$+RIGHT$(TM$,11-CP) :REM*162
340 PRINTMID$(TM$,CP,1);:CP=CP+1+11*(CP=11) :REM*50
350 IFCP/3=INT(CP/3)THENPRINT"{CRSR RT}";:CP=CP+1 :REM*174
360 IFCP=1THENPRINT"{11 CRSR LFs}"; :REM*54
370 GOTO250 :REM*189
380 POKE56335,CB :REM*205
390 AP$=RIGHT$(TM$,1):H=VAL(MID$(TM$,1,1))*16+VAL(MID$(TM$,2,1)
) :REM*232
400 M=VAL(MID$(TM$,4,1))*16+VAL(MID$(TM$,5,1)) :REM*31
410 S=VAL(MID$(TM$,7,1))*16+VAL(MID$(TM$,8,1)) :REM*153
420 HP=-128*(AP$="P")+H:POKE56331,HP :REM*181
430 POKE56330,M:POKE56329,S:POKE56328,VAL(MID$(TM$,10,1)) :REM*96
440 IFB$="2"THENPOKE56333,136 :REM*110
450 GOTO160 :REM*11

```

**Listing 6. Interrupt-generator routine for Listing 3.**

```

;NMI ROUTINE DEMO
;USING TIMERS A & B OF CIA2
;
EXP
LDA #<NEWNMI ;Change RAM NMI Vector
STA $0318
LDA #>NEWNMI
STA $0319
LDA #FFF ;Set up CIA2 registers
STA $DD04 ;Timer A LO
STA $DD05 ;Timer A HI
LDA #2F
STA $DD06 ;Timer B LO
LDA #00
STA $DD07 ;Timer B HI
LDA #51
STA $DD0F ;CRB - Timer B counts Timer A
LDA #11
STA $DD0E ;CRA - Timer A counts clock cycles
LDA #82
STA $DD0D ;ICR - Enable Timer B interrupts
RTS
NEWNMI
SEI ;New NMI routine: Set
;interrupt disable
PHA ;Save Registers .A, .X, ,
;and .Y to stack
TXA
PHA
TYA
PHA
LDA $DD0D ;Check ICR to see if interrupt
;is from CIA2
BPL RESTORE ;If not got to KERNAL routine
LDY #0F ;Rotate direction vectors of the balls
LDX #0E
LDA $COD7 ;Save last value to stack
PHA
DLOOP
LDA SD,X ;Push other values up one
STA SD,Y
DEY ;Do next value

```

will find that you can calculate the time it takes in seconds for a counter to reach zero by taking the latch value (high byte \* 256 + low byte) and dividing by 1,022,730.

Each of the counters also has a corresponding control register, CRA (\$0E) and CRB (\$0F). These are located at \$DC0E-\$DC0F (56334-56335) for CIA 1, and \$DD0E-\$DD0F (56590-56591) for CIA 2. Figure 6 outlines how these registers are set up.

These registers control other aspects of the CIA chip besides timers A and B: namely, the TOD clock (bit 7 of CRB) and the serial port. The only bits that are of concern in using the timers to generate interrupts are 0, 3, 4 and 5 of CRA and CRB, and bit 6 of CRB.

Bit 0 starts (1) or stops (0) the corresponding timer. Bit 3 selects One-Shot (1) or Continuous mode (0) as discussed earlier. Bit 4 forces the value in the time latch to be loaded into the corresponding timer.

Bit 5 of CRA selects whether timer A is counting microprocessor cycles or CNT pulses. Bits 5 and 6 of CRB determine what timer B will count. Figure 6 explains which values select which option for timer B.

Listing 6 is a machine language routine that sits at the end of the ball-animation program (Listing 3 in Part 1). It uses timers A and B on CIA 2 to generate an interrupt approximately every three seconds.

To do this, timer A latch is loaded with \$FFFF and timer B latch with \$002F. The control registers are set up so that timer A counts clock pulses and timer B counts timer A. The force-latch and start-timer bits are also set. The ICR is set to enable timer B interrupts.

At this time, NEWNMI is executed. An SEI instruction sets the interrupt disable flag in the processor status register. The registers are then pushed onto the stack (lines 2240-2280), and the ICR is checked to make sure that the interrupt came from CIA 2 (lines 2290-2300.) The 16 direction vectors for the eight balls are then shifted over one, causing the balls to travel in different directions.

Go through the program carefully to see how the CIA registers are being programmed, especially CRA and CRB. Try to calculate the exact time interval by using the formula stated above.

If you're using an assembler, compile the ball-animation program with Listing 6 added on. Note that line 2020 in Listing 3 is replaced with the new program lines. (Be sure first to save an unchanged copy of Listing 3, because Part 3

## PROGRAMMING

of this article will modify it again, in a different way.)

Then type in SYS 49152 and watch what happens. Note that even though the NMI interrupt is being used, pressing the restore key has no effect because the interrupt routine checks to make sure that timer B of CIA 2 is the source of the interrupt.

**Serial Port**—The serial port is accessed through registers SP (\$0C) and CRA (see Figure 6). When you're sending data, timer A is also used to set the baud rate.

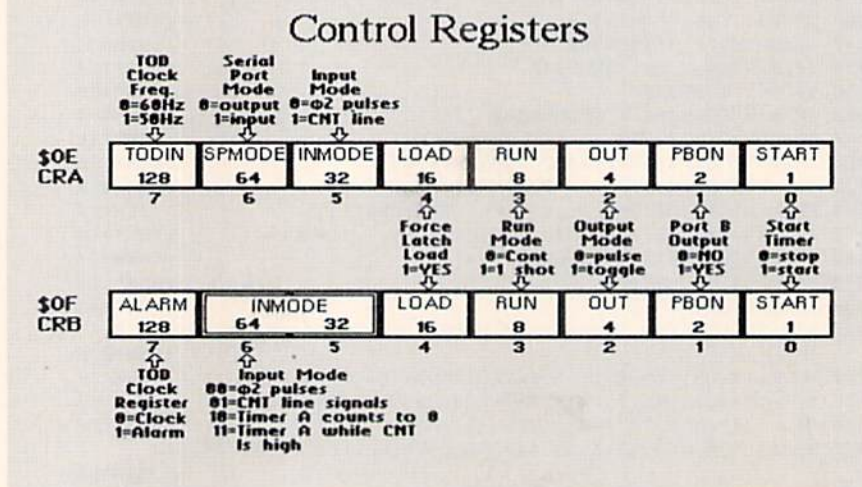
For more information on programming and using the CIA chip's serial port, consult the *Commodore 64 Programmer's Reference Guide*, Commodore Business Machines, Inc.; *Mapping the Commodore 64*, by Sheldon Leemon, COMPUTE! Books; and *GEOS Inside and Out*, by M. Tornsdorf and R. Kerkloh, Abacus Software, Inc., 1987.

**Flag Line**—Although CIA 1's flag line is already used internally by the 64's serial bus, CIA 2's flag line is available—on the user port at pin B—for whatever applications you can dream up. ☐

```

DEX
BPL DLOOP ;Check if done
PLA ;Get last value from stack
STA SD ;Put at the beginning
JMP $FEBC ;Jump out through KERNAL NMI exit
RESTORE
JMP $FE4C ;Jump to RESTORE key routine
    
```

Figure 6. Control registers.



## PART 3

### RECAP

So far, you have learned what interrupts are, how the C-64's operating system uses them and how you can use them. We have explored the CIA chips and how you can program them to produce interrupt requests from a variety of sources.

We will now explore perhaps the most fascinating chip inside this tiny eight-bit computer, the Video Interface Controller (VIC-II) chip.

In addition to its graphics and sprite capabilities, the VIC-II can also generate IRQ interrupts from four possible sources, creating many unique and interesting video effects, such as split text and graphics screens, more than eight sprites at once and multiple border colors. Many commercial programs and games make use of these remarkable effects to produce seemingly impossible video displays.

### THE VIC-II CHIP

Like the CIA chips, the VIC-II is a specialized peripheral processor unit (PPU) that generates the C-64's video display—whether text, graphics, sprites or any combination of them. Unlike the CIA, the VIC-II can directly address the RAM memory, although only 16K at a time. The RAM stores the text and bit-

mapped graphics screens, color memory, character definitions and sprite definitions.

The VIC-II also uses 47 internal registers, which let you program the VIC-II and also communicate with the 6510 microprocessor. Table 2 briefly describes the registers, only a few of which are actually needed when dealing with interrupts, but all are important in generating the video effects.

Interrupts from the VIC-II chip appear on the 6510's IRQ or maskable interrupt line. In many ways, programming the VIC-II to produce interrupts is similar to using the CIA chips; however, there are some important differences.

### WHERE VIC-II INTERRUPTS ORIGINATE

As mentioned earlier, there are four sources of interrupt requests from the VIC-II. Figure 7 shows where they originate. The first three relate to the actual video display, while the fourth deals with light pen data.

The first interrupt is the raster compare, which occurs when the current screen or raster line being displayed equals the value set in the raster-compare register.

Raster lines refer to the horizontal scan lines that make up the video dis-

play. To better understand them, you need to know a little bit about how your video display works. In order to create the text and graphics you see on the screen, an electron beam produced by the cathode ray tube (CRT) of the video display scans a total of 262 horizontal lines across the screen.

If you look closely, you can see the individual scan lines or raster lines. Although there are 262 lines on an American NTSC standard display screen, the VIC-II uses only 200 of them—lines 50 through 249—to create the text or graphics display. The rest of the lines make up the upper and lower borders.

In their trip across the screen, the scan lines are divided up into pixels making up the left border, 320 dots of horizontal resolution, then the right border. The screen display starts at pixel 24. Every one of these lines is updated 60 times a second.

It's similar to building a brick wall, 262 layers thick from top to bottom, every sixtieth of a second. By using different colored bricks in the proper locations, you can create a text or bit-mapped graphics screen, depending on where the screen-display information is coming from.

In the animation demonstrations we used earlier, you may have noticed some flickering of the sprites as they traveled

## PROGRAMMING

around the screen. This happens when the interrupt to change their location occurs when a sprite is only halfway displayed, or while the MSB register is being altered. You can avoid this flickering by selecting a raster interrupt to change the positions while the electron beam is not on the display area. I'll demonstrate this use of the raster-compare interrupt later.

The second and third types of interrupts have to do with the sprites, or Movable Object Blocks (MOBs), as Commodore calls them. The VIC-II chip can detect when there is a collision between a sprite and the bitmapped graphics display, or between two sprites. Although a sprite is defined by a 24-by-21 rectangular grid, collisions only occur when a sprite's dot touches another sprite or a text character or part of the bitmapped graphics screen.

The fourth source of VIC-II interrupts is the light pen—a pen-shaped probe with a light-sensing device in the tip that is placed directly on the video display. When the scan line passes under the light sensor, the location of the light pen is stored in two of the VIC-II registers and an interrupt is generated.

The light-pen input is connected to the fire-button of control port 1. Therefore, this interrupt can also be activated by pressing the fire-button of a joystick plugged into the control port. In this case, the values of the light pen location registers would be meaningless.

### HOW THESE INTERRUPTS GET TO THE 6510

Like the CIA chips, all four sources of interrupts from the VIC-II are controlled by an internal register. Unlike the CIA chip, this is done through two registers instead of one. These are registers \$19-\$1A, which are located at \$D019-\$D01A (53273-53274). Figure 8 shows how they are set up.

The first is the interrupt flag register, which, like the ICR register of the CIA chip, signals when an interrupt has originated from the VIC-II chip and indicates which of the possible sources has generated the interrupt request. As shown in Figure 8, bit 7 will be set when any of the four sources generates an interrupt. The corresponding bits of the sources responsible will also be set.

When an interrupt occurs from a particular source, a latch, which prevents that interrupt from occurring again until the latch is cleared, is also set. This is done by writing 1 to the appropriate bit of the interrupt flag register.

The second register is the interrupt-enable register, which selects or enables

a particular interrupt by setting the appropriate bit to 1. To clear or disable an interrupt, the appropriate bit must be set to 0.

For example, in a program to determine when a missile hits a spaceship, the interrupt for a sprite-to-sprite collision is enabled by writing a value of \$04 to the interrupt-enable register

(\$1A). When the missile touches the spaceship, an IRQ interrupt will occur, and bits 7 and 2 of the interrupt flag register are set.

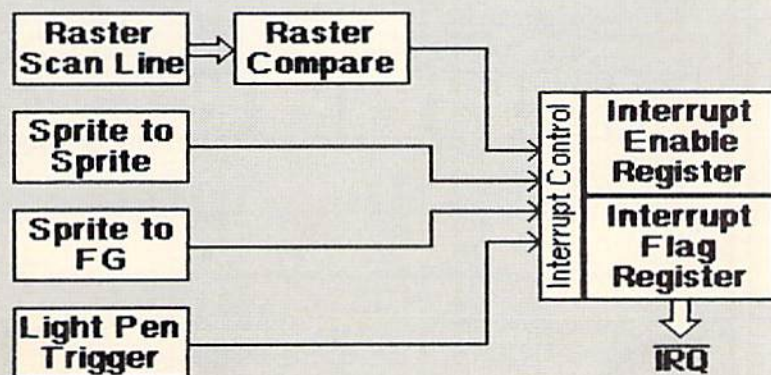
In addition, the appropriate bits of the sprite-to-sprite collision register (\$1E) are set. This register, which is located at \$D01E (53278), keeps track of which sprites are involved in a collision. ▶

Table 2. Video interface controller register map.

Register	Address	Description
MOBXY	\$00-\$0F	Sprite position registers (0X, 0Y, 1X,...,7Y)
MSIGX	\$10	Most significant bit of X-position registers
CNTY	\$11	* Control register (bit 8 of raster-compare register)
RASTER	\$12	* Read-raster scan line/write-raster compare register
LPX	\$13	* Light pen X position (0-160)
LPY	\$14	* Light pen Y position (0-199)
MOBEN	\$15	Sprite enable register
CNTX	\$16	Control register
MOBYEX	\$17	Sprite vertical expansion register
MCR	\$18	Memory control register
INTFLAG	\$19	* VIC-II interrupt flag register
INTEN	\$1A	* VIC-II interrupt enable register
MOBPR	\$1B	Sprite-to-foreground display priority
MOBMC	\$1C	Sprite multicolor mode enable register
MOBXEX	\$1D	Sprite X-expand register
MOBMOB	\$1E	* Sprite-to-sprite collision register
MOBFG	\$1F	* Sprite-to-foreground collision register
EXTCOL	\$20	Border color
BGCOL0	\$21	Background color 0
BGCOL1	\$22	Background color 1, multicolor 1
BGCOL2	\$23	Background color 2, multicolor 2
BGCOL3	\$24	Background color 3, multicolor 3
MOBMC0	\$25	Sprite multicolor 0
MOBMC1	\$26	Sprite multicolor 1
MOBCOL	\$27-\$2E	Sprite color registers

\* Registers that are important in VIC-II interrupts.

Figure 7. Sources of interrupt requests from the VIC-II.



## PROGRAMMING

If two sprites make contact, the corresponding bits of the collision register are set to 1. A similar register (\$1F) is used to monitor sprite-to-foreground collisions (see Figure 9).

Also, the sprite-to-sprite collision interrupt latch will be set, preventing any further sprite collisions from generating an interrupt. After the collision is handled by the interrupt routine, this

latch should be cleared by writing a \$04 to the interrupt flag register.

### HOW THESE INTERRUPTS ARE USED

**Raster-compare interrupt**—Using raster-compare interrupts involves setting the raster-compare register to the appropriate value, enabling the interrupt using the interrupt-enable register and installing an interrupt handling routine by changing the IRQ RAM vector (\$0314-\$0315). Listing 7 demonstrates how the raster-compare interrupt can be used to create an unusual video effect.

Essentially, the screen is split into two halves. The bottom half is normal text. The top half displays text that waves back and forth by means of the horizontal fine-scroll register, while text at the bottom of the screen remains stationary.

The split is accomplished by setting a raster interrupt to occur at scan line 0, then again at line 154. The first interrupt loads the next value from the fine-scroll data at the end of the program and stores it in the horizontal fine-scroll and control register (\$16), then sets the raster-compare register to generate the next interrupt at line 154. At this time, the fine scroll is set to 0, as is the raster-compare register.

Since the raster-compare register is only eight bits and there are 262 possible scan lines, a ninth bit must be used to select raster lines greater than 255. This bit is located in VIC-II register \$11, bit 7 (see Figure 10).

The raster-compare register has a dual function. When it is read, it will return the value of the current scan line. This is useful to help avoid flickering of the screen display while areas of the bitmapped graphics screen are being changed. Make sure that you also check bit 7 of register \$11 as well.

Examine Listing 7 carefully. It is important to understand that two raster-compare interrupts must occur during every update of the screen to accomplish the desired effect. Enter and assemble the program. To run it, just type in SYS 49152; then watch.

Listing 7 also demonstrates another use for the raster-compare interrupt. Essentially, instead of using timer A of CIA 1 to generate the sixtieth-of-a-second housekeeping interrupt, this now occurs at scan line 154. This works out fine, since the screen is also updated 60 times a second.

Lines 1150-1220 set up the raster interrupt and disable the interrupt from timer A of CIA 1. Its major effect is

Figure 8. The interrupt-flag and interrupt-enable registers.

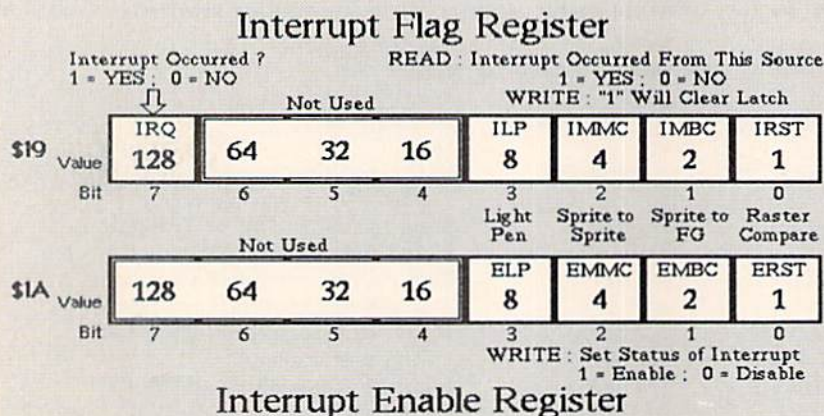


Figure 9. The sprite-to-foreground and sprite-to-sprite collision registers.

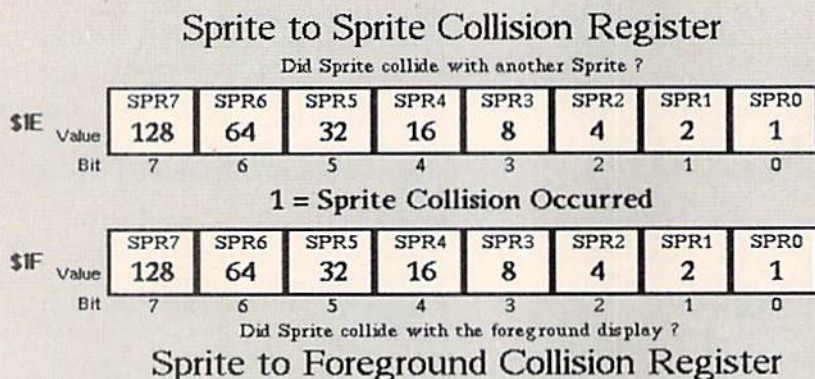
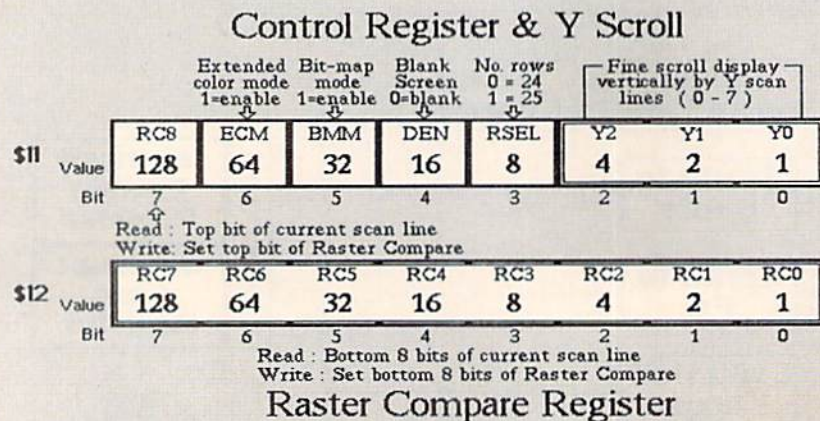


Figure 10. The Y scroll and control and raster-compare registers.



## PROGRAMMING

eliminating any flicker that may occur by having the timer A interrupt occur slightly before the raster-compare interrupt. When this happens, the raster-compare interrupt will not occur, since the interrupt-disable flag of the 6510 control register is set. This technique, which is useful in generating animation with sprites, is used in Listing 8.

**Sprite-to-foreground and sprite-to-sprite collisions**—Both of these interrupts are useful when you write game programs that use animated sprites. They let you detect when a sprite touches another sprite or the foreground—whether it be character graphics or bitmapped graphics—so you can take the appropriate action.

Although there are eight sprites, only two collision-detection interrupts exist. To determine which sprite is involved in the collision, the VIC-II uses two registers for sprite-to-sprite and sprite-to-foreground collisions (\$1E and \$1F, respectively). Each bit of the register corresponds to one of the eight sprites, as shown in Figure 10. By checking these registers, you can determine which of the sprites are involved in the collision.

However, reading these registers will clear their contents, so a copy should be saved if it is needed. These registers should be cleared before you expect a collision interrupt to occur, so that the collisions will be accurately detected.

Listing 8 is yet another modification to the ball-animation demo (Listing 3 in Part 1). Like Listing 6 in Part 2, it is designed to sit at the end of Listing 3 and uses the sprite-to-foreground collision interrupt to determine when a ball collides with a border set up around the screen by Listing 9. When a collision occurs, the ball's direction is changed, and it appears to be bouncing off the wall.

Note that lines 1390 through 1410 of the original Listing 3 will need to be changed to a JMP instruction. The new line should read:

```
1390 C04C 4C 30 C1 NEWIRQ
      JMP ICHECK
```

This lets the IRQ routine determine whether the interrupt is from the raster compare or from a collision. Line 2020 is replaced by the code in Listing 8, which sets up the VIC-II to generate the housekeeping interrupt at scan line 0 and contains the sprite-to-foreground collision-handling routine.

Since the animation and the collision both use the IRQ interrupt generated from the VIC-II chip, the program must determine which interrupt it is servicing by checking the interrupt flag reg-

**Listing 7. An unusual visual effect using the raster-compare interrupt.**

```
*= $C000
;
; THE SHAKING SCREEN
; ***** RASTER COMPARE DEMO *****
;
      SEI                      ;Set Interrupt Disable Flag
      LDA #<NEWIRQ            ;Change IRQ RAM Vector
      STA $0314
      LDA #>NEWIRQ
      STA $0315
      CLI                      ;Clear Interrupt Disable Flag
      LDA #$00                ;Initialize counters and VIC-II chip
      STA $02                  ;Set pointer to zero
      STA $D012                ;Set Raster Compare for scan line 0
      LDA #$1B
      STA $D011                ;Set top bit to 0
      LDA #$01
      STA $D01A                ;Enable Raster Compare Interrupt
      STA $DC0D                ;Disable Timer A interrupt from CIA 1
      RTS
NEWIRQ
      LDX $D012                ;Get value of current scan line
      BNE NORMAL              ;If not at scan line 0 normal display
SHIFT
      LDY $02                  ;Get pointer to fine scroll data
      INY                      ;Increment pointer
      CPY #$0E                 ;If at end of data ...
      BNE GETY
      LDY #$00                 ;... then start again
GETY
      LDA YS,Y                 ;Get fine scroll data
      STA $D016                ;Put in horizontal fine scroll
                                ;register
      STY $02                  ;Save pointer
      LDA #$01
      STA $D019                ;Reset Raster Compare Interrupt Latch
      LDA #$9A                 ;Load value of next Raster
                                ;Compare (154)
      STA $D012                ;Save to Raster Compare Register
      JMP $EA81                ;Exit interrupt routine through
                                ;the KERNAL
NORMAL
      LDA #$08                 ;Set fine scroll to 0
      STA $D016                ;Save to horizontal fine
                                ;scroll register
      LDA #$01
      STA $D019                ;Reset Raster Compare Interrupt Latch
      LDA #$00                 ;Load value of next Raster Compare (0)
      STA $D012                ;Save to Raster Compare Register
      JMP $EA31                ;Jump to KERNAL IRQ routine
YS
      .BYTE 8 9 10 11 12 13 14 15 ;Fine scroll data
      .BYTE 14 13 12 11 10 9
```

**Listing 8. Demo that detects when a ball collides with a border.**

```
;MODIFIED FLYING BALLS - BOUNCING BALLS
;WITH RASTER COMPARE GENERATED INTERRUPT
;AND SPRITE TO FOREGROUND COLLISION DETECTION
;
; EXP
      LDA #$1B                 ;Initialize VIC-II registers
      STA $D011                ;Control Register - 8th bit of
                                ;Raster Compare
      LDA #0
      STA $D012                ;Raster Compare Register
      LDA #$03
      STA $D01A                ;Interrupt Enable - ERST and EMBC
      STA $DC0D                ;Turn off Timer A interrupt from CIA 1
      RTS
ICHECK
      LDA $D019                ;Get Interrupt Flag Register
      AND #$02                 ;Check if IMBC has occurred
      BNE COLLISION           ;If YES - then go to collision routine
      LDA #$01                 ;If NO - Turn off Raster Compare Latch
      STA $D019
      LDA $D01F                ;Clear Sprite - Foreground
                                ;Collision Register
      JMP CONT                 ;Jump to ball animation routine
COLLISION
      LDA $D01F                ;Get Sprite to Foreground
```

# PROGRAMMING

```

                                ;Collision Register
                                ;Set Y to 7 - Sprite Number Pointer
LDY #$07
LDX #$00                                ;Set X to 0 - Sprite Collision Pointer
SHIFT
ASL                                    ;Check if collision occurred
                                ;with Sprite Y
BCC NOCOLL                            ;If NO - try next Sprite
PHA                                    ;If YES -
TYA
STA CSPRITE,X                          ;Save Sprite Number
PLA
INX                                    ;Increment Sprite Collision Pointer
NOCOLL
DEY                                    ;Decrement Sprite Number Pointer
BPL SHIFT                              ;If not all sprites checked,
                                ;do next one
DEX                                    ;Point to last sprite
                                ;collision detected
NEXTA
LDA CSPRITE,X                          ;Get Sprite Number
TAY                                    ;Put SN in Y
PHA                                    ;Save SN to the Stack
LDA MASK,Y                             ;Get MSB mask for Sprite
STA $FE                                ;Store in zero page location $FE
PLA                                    ;Get SN back off stack
ASL                                    ;Multiply by 2
TAY                                    ;Put new pointer in Y
LDA $D000,Y                            ;Get X location of Sprite
PHA                                    ;Save to the Stack
CMP #$1C                               ;Check if collision is at left border
BCS NOTLEFT                            ;If not check right side
LDA $FE                                ;Get MSB mask - Check MSB
                                ;Register to make
                                ;sure Sprite is at the left side
AND $D010
BNE NOTLEFT                            ;If NO then check right side
LDA #0
STA SD,Y                               ;Change X direction of Sprite
                                ;Save to Sprite Direction data
NOTLEFT
PLA                                    ;Get X location back from stack
CMP #$3D                               ;Check if at right border
BCC NOTRIGHT                           ;If NO - check top of screen
LDA $FE                                ;Get MSB mask
AND $D010
BEQ NOTRIGHT                           ;Check MSB bit
                                ;If not at right - check top of screen
LDA #$80
STA SD,Y                               ;Change X direction of Sprite
                                ;Save to Sprite Direction data
NOTRIGHT
INY                                    ;Point to Y value of Sprite Location
LDA $D000,Y                            ;Get Y value of Sprite Location
CMP #$35                               ;Check if at top border
BCS NOTTOP                             ;If NO - then check bottom of screen
LDA #0
STA SD,Y                               ;Change Sprite Y Direction
NOTTOP
CMP #$E2                               ;Check if at bottom border
BCC NOTBOTTOM                          ;If NO - Leave
LDA #$80
STA SD,Y                               ;Change Sprite Y Direction
NOTBOTTOM
DEX                                    ;Decrement Sprite Collision Pointer
BPL NEXTA                              ;If more sprites - Check next sprite
LDA #$02
                                ;Clear Sprite to Foreground
                                ;Interrupt Latch
STA $D019
JMP $EA81                              ;Jump to KERNAL IRQ exit routine
CSPRITE
.BYTE 0 0 0 0 0 0 0 0 ;Sprites that collided
MASK
.BYTE 1 2 4 8 16 32 64 128 ;Mask values

```

## Listing 9. Creating the border for the demo.

```

10 REM(2 SPACES)SCREEN BORDER SET UP FOR BOUNCING BALLS:REM*251
20 REM                                :REM*163
30 IFA=0 THEN PRINT "{SHFT CLR}LOADING BBALLS.IRQ":A=1:LOAD"BBALL
S.IRQ",8,1                                :REM*140
40 PRINT "{SHFT CLR}{CTRL 2}{COMD A}{38 SHFT *s}{COMD S}";
                                :REM*16
50 FORX=1TO23:PRINT "{SHFT -}{38 SPACES}{SHFT -}";NEXT :REM*82
60 PRINT "{COMD Z}{38 SHFT *s}{HOME}";:POKE2023,125 :REM*168
70 SYS49152                                :REM*244
80 GOTO80                                :REM*216

```

ister. This occurs in lines 2160-2180.

Enter Listing 8 and assemble it. Save a copy under the filename BBALLS.IRQ, then load up Listing 9 and type in RUN. The rest is magic. For a further exercise in working with VIC-II interrupts, try to modify the program to detect collisions between balls, and send them bouncing off each other as well.

**Light pen interrupts**—The final source of interrupts is the light pen. However, as discussed earlier, this interrupt can also be triggered by pressing the fire-button of a joystick plugged into control port 1.

When you use a light pen, the values of registers \$13-\$14 contain the screen position of the pen. These registers are located at \$D013-\$D014 (53267-53268). The horizontal position register (\$13) is too small to hold all the possible values corresponding to the 320 horizontal pixel positions, so the VIC-II chip compromises by returning a value from 0 to 159, or every other pixel position.

With the proper amount of creativity, the possible effects that can be created with interrupts are endless. The ball-animation demo is just a simple example of what can be accomplished by accessing the hidden powers of the VIC-II video display chip. The rest is up to you.

## EXTERNAL INTERRUPTS

Throughout this article, I've introduced several ways of creating interrupts from outside the C-64.

The first is to directly connect to the IRQ and NMI lines of the 6510 microprocessor through the expansion port. Second, you can access the flag line of CIA 2 on the user port, which generates interrupt requests on the NMI line. This method has the advantage of being able to disable the interrupt via the ICR register, if desired. The third method is through the fire-button of the joystick 1 port, which activates the light-pen interrupt of the VIC-II chip, causing an interrupt request on the IRQ line. It also can be turned off and on by setting the appropriate bit of the interrupt-enable register.

The choice is yours as to what you actually connect up to generate these interrupts. This article contains all the fundamentals for generating and programming with interrupts. But the best way to learn about them is to play around with them yourself. ■

*Jim Hosek, who has been programming for some ten years, is a veterinarian attached to the Veterinary Hospital of the University of Pennsylvania in Philadelphia.*



# LEARN TO WALK BEFORE YOU RUN

WE RECEIVE MANY LETTERS from new Commodore owners who want to type in program listings from *RUN* and need help in getting started. To answer many of the questions novice users have, we present the following guidelines.

1. First, keep in mind that as a beginner you should enter only short Basic programs. Avoid machine language listings and lengthy Basic programs until you get the hang of what you're doing.
2. To help you catch mistakes in typing in listings, we publish *RUN*'s Checksum program elsewhere in this issue.
3. If you intend to save the program you're typing in on a brand new disk, you must format that disk. To do this, insert the disk in your drive and type:

```
OPEN15,8,15 <press return>  
PRINT#15,"N0:NAME,##" <press return>
```

The ## is a two-character identifier that can be any combination of letters or digits. NAME can be any title for the disk that you choose, as long as it's 16 characters or less.

After entering the above lines, wait for a few minutes while the disk spins inside the drive. When the disk stops, the formatting is done. Then type:

```
CLOSE15 <press return>
```

In 128 mode on a C-128, you can shorten this procedure by typing:

```
HEADER "NAME,##" <press return>
```

**Caution:** The formatting process erases any material already on the disk, so if you're formatting a used disk, make sure it doesn't contain any programs you want to keep. See item 11 below, on reading the disk directory, if you need to find out what's on the disk.

4. Before you start typing in a program listing, your computer's memory needs to be empty. To make sure it is, turn the computer off, wait a few seconds, and turn it on again.
5. As you type in the listing, remember to press the return key after typing each line. This enters the line into memory.
6. If you want to review what you've entered, type LIST and press the return key; all the lines you've entered will scroll by. You can slow the scrolling on the C-64 by holding down the control key, and on the C-128 by pressing the no-scroll key. To view certain specific lines, type LIST, followed by the line numbers you want; then press the return key. For example, LIST 10-50 displays lines 10 through 50, and LIST 20 displays only line 20.
7. If you find an error in a line, delete the incorrect characters with the insert-delete key, then retype that portion and press return to enter the new line in memory.

8. Be sure to save the program to disk fairly often during the typing process. Otherwise, you could lose all your work if a power glitch wipes out your computer's memory. To save a partial or complete Basic program listing, type:

```
SAVE "NAME",8 <press return>
```

In 128 mode on a C-128, you can press F5, type in NAME and press the return key. Here, NAME is the filename you want the program to have, not the disk name you used when formatting.

Each time you save a revised program to the same disk, you must change its filename, or a disk error will occur, even if only one character is changed. An easy way to vary the filename is by adding version numbers to the end of the basic name (Program.1 and Program.2, for example). The numbers will also tell you which version is the latest.

9. If you wish to erase (scratch) unwanted programs from a disk, type:

```
OPEN15,8,15 <press return>  
PRINT#15,"S0:filename" <press return and wait a few seconds>  
CLOSE15 <press return>
```

In 128 mode on a C-128, you can type:

```
SCRATCH "NAME" <press return>
```

*Be sure not to erase the final version!*

10. Always save the final version of a program to two disks, so you have a backup copy in case one of the disks gets damaged. When saving to two different disks, you can use the same program name in each case.
11. To view a complete list of the filenames on a disk (i.e., read the disk directory), type:

```
LOAD "$",8 <press return>  
LIST <press return>
```

In 128 mode on a C-128, you can just press F3.

12. When you know what program you want to load, next make sure you know *exactly* how its filename is spelled in the disk directory, including punctuation, special characters and spaces. A mistake in the filename will keep the load from working.

If the disk directory is still on the screen when you enter the Load command, you can refer to that for the spelling. If the directory will be gone from the screen by the time you enter the Load command, jot down the exact spelling of the filename for reference. Once you're sure of the filename, load the program by typing:

```
LOAD "NAME",8 <press return>
```

In 128 mode on a C-128, you can just press F2, type in the filename and press the return key.

13. After you've loaded the program, enter RUN to use it. ■



# Tag 'Em

*If your goal is to have yourself a ball, why not play Tag?*



By JOHN FEDOR


**J**ust grab the ball and run into the goal. That's all you have to do to score in Tag. Sound easy? Well, watch out! Your opponent tries to stop you—and to make goals of his own.

If your opponent tags you while you have the ball, you momentarily freeze in place. To prevent this, release the ball by pressing the fire-button before he tags you. The ball then continues

moving in the direction it was going when released.

As you play, beware of the center, and keep in mind that the corners can be a little "sticky." The winner is the first player to reach five points, but you probably won't even notice how the scores are adding up, since you'll be having so much fun trying to "burn" your opponent.

The joystick in port 1 controls the

player who is defending the goal on the left of the screen, and the joystick in port 2 controls the player defending the right-hand goal. Play begins when either gamer presses his fire-button. The music that accompanies Tag was composed by Ken Fountain. 

*John Fedor is a college student with an avid interest in computers and math. On rainy days, he plays tag on his Commodore.*

## Listing 1. Tag program.

```

0 REM TAG - JOHN FEDOR :REM*34
10 POKE 53280,12:POKE 53281,1 :REM*78
20 PRINT"{SHFT CLR}{CTRL 1}{2 C
   RSR DNs}"TAB(19)"A" :REM*58
30 FORX=1TO17:PRINT"{HOME}{2 CR
   SR DNs}"TAB(X)" T"TAB(37-X)"
   G " :REM*153
40 FORQ=1TO25:NEXTQ,X :REM*145
50 PRINT"{HOME}{5 CRSR DNs}"TAB
   (15)"JOHN FEDOR" :REM*16
60 PRINT"{18 CRSR DNs}{4 SPACES
   }PLEASE WAIT...LOADING IN DA
   TA...{HOME}" :REM*199
70 FORX=49152TO51711:READA:POKE
   X,A:CK=CK+A:NEXTX :REM*245
80 IF CK<>231639 THEN PRINT "ER
   ROR IN DATA STATEMENTS...":E
   ND :REM*194
90 SYS 49152 :REM*1
100 DATA 169,48,141,60,3,141,61
   ,3,32,214,192,76,155,194,16
   2,0,189,244,199 :REM*50
110 DATA 32,210,255,232,224,5,2
   08,245,141,32,208,141,33,20
   8,133,251,133 :REM*119
120 DATA 253,169,4,133,252,169,
   216,133,254,162,0,160,0,169
   ,1,145,253,169 :REM*216
130 DATA 160,145,251,160,39,169
   ,160,145,251,169,1,145,253,
   24,165,251,105 :REM*190
140 DATA 40,133,251,133,253,165
   ,252,105,0,133,252,105,212,
   133,254,232,224 :REM*99
150 DATA 11,144,212,224,13,144,
   228,224,25,144,204,162,0,16
   9,160,157,0,4 :REM*165
160 DATA 157,192,7,169,1,157,0,
   216,157,192,219,232,224,40,
   208,235,162,0 :REM*235
170 DATA 189,238,198,157,128,63
   ,232,224,128,208,245,169,0,
   133,251,169,10 :REM*97
180 DATA 133,252,24,166,252,160
   ,12,32,240,255,166,251,160,
   0,189,110,199,32 :REM*157
190 DATA 210,255,232,200,192,16
   ,208,244,24,165,251,105,16,
   133,251,230,252 :REM*173
200 DATA 201,64,208,219,169,7,1
   41,184,217,141,224,217,141,
   223,217,141,7 :REM*140
210 DATA 218,32,27,194,173,60,3
   ,9,48,141,209,5,173,61,3,9,
   48,141,249,5,96 :REM*95
220 DATA 162,0,138,157,0,212,23
   2,224,24,208,248,169,15,141
   ,24,212,169,112 :REM*100
230 DATA 141,6,212,169,240,141,
   13,212,169,17,141,19,212,16
   2,0,142,85,3,142 :REM*211
240 DATA 86,3,142,87,3,142,91,3
   ,142,92,3,142,93,3,232,142,
   88,3,142,89,3 :REM*172
250 DATA 142,90,3,142,94,3,120,
   173,20,3,133,113,173,21,3,1
   33,114,169,193 :REM*167
260 DATA 141,21,3,169,45,141,20
   ,3,88,96,206,94,3,173,94,3,
   240,3,108,113,0 :REM*217
270 DATA 169,8,141,94,3,206,88,
   3,173,88,3,208,56,173,91,3,
   208,5,169,32,141 :REM*197
280 DATA 4,212,238,85,3,174,85,
   3,189,249,199,133,7,189,99,
   200,41,128,141 :REM*119
290 DATA 91,3,189,99,200,41,127
   ,141,88,3,166,7,189,206,200
   ,141,0,212,189 :REM*6
300 DATA 219,200,141,1,212,169,
   33,141,4,212,206,89,3,173,8
   9,3,208,56,173 :REM*23
310 DATA 92,3,208,5,169,16,141,
   11,212,238,86,3,174,86,3,18
   9,232,200,133,7 :REM*177
320 DATA 189,51,201,41,128,141,
   92,3,189,51,201,41,127,141,
   89,3,166,7,189 :REM*23
330 DATA 206,200,141,7,212,189,
   219,200,141,8,212,169,17,14
   1,11,212,206,90 :REM*19
340 DATA 3,173,90,3,208,56,173,
   93,3,208,5,169,128,141,18,2
   12,238,87,3,174 :REM*181
350 DATA 87,3,189,127,201,133,7
   ,189,186,201,41,128,141,93,
   3,189,186,201,41 :REM*132
360 DATA 127,141,90,3,166,7,189

```

RUN it right: C-64; Two joysticks

# G A M E S

	,206,200,141,14,212,189,219 200,141,15,212 :REM*200	8,15,192,120,144,11,192,155 176,7,169,1 :REM*174	208,152,157,1,208,165,253,7 3,255,45,16 :REM*162
370	DATA 169,129,141,18,212,173 ,85,3,201,106,208,20,162,15 ,142,24,212,202 :REM*219	600 DATA 141,62,3,230,21,192,12 0,208,15,224,108,144,11,224 ,239,176,7,169 :REM*168	830 DATA 208,5,2,141,16,208,173 ,68,3,197,254,208,35,165,25 1,41,16,240,29 :REM*64
380	DATA 208,250,120,165,113,14 1,20,3,165,114,141,21,3,88, 108,113,0,162,97 :REM*55	610 DATA 0,141,63,3,230,21,192, 154,208,15,224,108,144,11,2 24,239,176,7 :REM*218	840 DATA 166,254,169,32,157,65, 3,189,39,208,9,8,157,39,208 ,162,32,142,11 :REM*243
390	DATA 160,225,173,68,3,201,1 ,208,2,162,32,201,2,208,2,1 60,32,142,223,5 :REM*137	620 DATA 169,1,141,63,3,230,21, 76,223,195,224,66,208,7,169 ,0,141,62,3 :REM*221	850 DATA 212,232,142,11,212,169 ,0,141,68,3,96,162,2,160,0, 136,208,253 :REM*202
400	DATA 142,7,6,140,184,5,140, 224,5,96,174,4,208,224,27,2 08,34,173,16,208 :REM*186	630 DATA 230,21,165,21,240,9,16 2,32,142,4,212,232,142,4,21 2,96,206,69,3 :REM*95	860 DATA 202,208,250,96,173,20, 3,133,113,173,21,3,133,114, 120,169,197 :REM*0
410	DATA 41,4,208,27,206,4,208, 206,0,208,32,128,197,173,4, 208,201,18,208 :REM*35	640 DATA 173,69,3,240,1,96,169, 2,141,69,3,173,1,220,41,31, 73,31,133,251 :REM*157	870 DATA 141,21,3,169,162,141,2 0,3,88,96,32,15,195,32,27,1 94,162,0,222 :REM*181
420	DATA 240,238,61,3,104,104,3 2,142,194,76,155,194,174,2, 208,224,65,208 :REM*72	650 DATA 162,2,134,252,134,253, 202,134,254,32,120,196,173, 68,3,201,1,208 :REM*36	880 DATA 64,3,189,64,3,201,255, 208,5,169,0,157,64,3,232,22 4,4,208,236 :REM*73
430	DATA 34,173,16,208,41,2,240 ,27,238,2,208,238,0,208,32, 128,197,173,2 :REM*90	660 DATA 34,174,2,208,172,3,208 ,142,0,208,140,1,208,162,0, 173,16,208,41 :REM*107	890 DATA 173,66,3,208,8,173,40, 208,41,7,141,40,208,173,67, 3,208,8,173,41 :REM*42
440	DATA 208,201,72,208,240,238 ,60,3,104,104,32,142,194,76 ,155,194,96,120 :REM*67	670 DATA 2,240,1,232,134,2,173, 16,208,41,254,5,2,141,16,20 8,173,0,220,11 :REM*142	900 DATA 208,41,7,141,41,208,17 3,30,208,133,20,165,20,41,6 ,201,6,240,35 :REM*58
450	DATA 165,113,141,20,3,165,1 14,141,21,3,88,96,169,5,205 ,60,3,240,5,205 :REM*225	680 DATA 31,3,31,133,251,169,4 ,133,252,133,253,74,133,254 ,32,120,196 :REM*230	910 DATA 165,20,201,3,208,10,17 3,66,3,208,5,169,1,141,68,3 ,165,20,201,5 :REM*223
460	DATA 61,3,208,3,32,214,192, 32,14,192,32,217,198,165,2, 41,16,208,247 :REM*69	690 DATA 173,68,3,201,2,208,34, 174,4,208,172,5,208,142,0,2 08,140,1,208 :REM*113	920 DATA 208,10,173,67,3,208,5, 169,2,141,68,3,108,113,0,17 3,68,3,240,248 :REM*7
470	DATA 162,0,189,174,199,32,2 10,255,232,224,40,208,245,2 4,162,24,160,5 :REM*12	700 DATA 162,0,173,16,208,41,4, 240,1,232,134,2,173,16,208, 41,254,5,2,141 :REM*35	930 DATA 162,32,142,11,212,232, 142,11,212,201,1,208,33,169 ,16,141,67,3 :REM*58
480	DATA 32,240,255,162,0,189,2 14,199,32,210,255,232,224,3 0,208,245,32,217 :REM*5	710 DATA 16,208,96,166,254,189, 63,3,240,3,76,85,197,173,16 ,208,37,253 :REM*132	940 DATA 169,48,141,66,3,169,32 ,141,64,3,169,10,141,40,208 ,169,14,141,41 :REM*214
490	DATA 198,165,2,41,16,240,24 7,173,60,3,201,5,176,7 :REM*139	720 DATA 133,2,166,252,188,1,20 8,189,0,208,170,165,251,41, 1,240,1,136 :REM*68	950 DATA 208,169,0,141,68,3,108 ,113,0,169,16,141,66,3,169, 48,141,67,3 :REM*204
500	DATA 173,61,3,201,5,144,6,3 2,172,198,32,142,194,32,14, 192,32,139,197 :REM*56	730 DATA 165,251,41,2,240,1,200 ,165,251,41,4,240,1,202,165 ,251,41,8,240 :REM*67	960 DATA 169,32,141,65,3,76,41, 198,169,0,170,157,64,3,232, 224,5,208,248 :REM*208
510	DATA 32,77,198,173,30,208,2 08,251,32,237,195,32,128,19 7,32,59,194,76 :REM*202	740 DATA 1,232,224,0,208,10,165 ,251,41,8,240,4,165,253,133 ,2,224,255,208 :REM*12	970 DATA 169,2,141,40,208,141,6 9,3,162,6,142,41,208,232,14 2,39,208,173,4 :REM*17
520	DATA 3,195,173,68,3,240,1,9 6,173,16,208,41,1,133,20,16 9,0,133,21,174 :REM*241	750 DATA 10,165,251,41,4,240,4, 169,0,133,2,192,56,176,1,20 0,192,227,144 :REM*63	980 DATA 220,41,1,141,62,3,173, 5,220,41,1,141,63,3,162,254 ,142,249,7,142 :REM*113
530	DATA 0,208,172,1,208,173,62 ,3,208,2,202,202,232,173,63 ,3,208,2,136 :REM*218	760 DATA 1,136,165,2,208,79,224 ,107,208,9,192,119,144,5,19 2,156,176,1 :REM*69	990 DATA 250,7,232,142,248,7,16 0,137,140,3,208,140,5,208,1 60,100,140,1 :REM*128
540	DATA 136,200,224,0,208,9,17 3,62,3,240,4,169,1,133,20,2 24,255,208,9 :REM*195	770 DATA 202,224,239,208,9,192, 119,144,5,192,156,176,1,232 ,192,119,208,9 :REM*215	1000 DATA 208,162,173,142,0,208 ,162,28,142,2,208,162,64,1 42,4,208,169,4 :REM*216
550	DATA 173,62,3,208,4,169,0,1 33,20,142,0,208,140,1,208,1 73,16,208,41,6 :REM*55	780 DATA 224,107,144,5,224,240, 176,1,136,192,155,208,9,224 ,107,144,5,224 :REM*221	1010 DATA 141,16,208,169,7,141, 21,208,96,169,0,141,60,3,1 41,61,3,162,0 :REM*124
560	DATA 5,20,141,16,208,192,54 ,208,7,169,1,141,63,3,230,2 1,192,228,208 :REM*26	790 DATA 240,176,1,200,224,28,1 76,20,192,134,144,15,192,14 1,176,11,173 :REM*255	1020 DATA 138,157,0,212,232,224 ,24,208,248,169,15,141,24, 212,169,50,141,5 :REM*175
570	DATA 7,169,0,141,63,3,230,2 1,165,20,208,90,224,26,208, 7,169,1,141,62 :REM*136	800 DATA 68,3,197,254,208,4,201 ,2,240,1,232,76,63,197,224, 65,144,20,192 :REM*201	1030 DATA 212,169,15,141,1,212, 169,57,141,12,212,169,5,14 1,8,212,96,173,0 :REM*222
580	DATA 3,230,21,224,108,208,1 5,192,120,144,11,192,155,17 6,7,169,0,141 :REM*72	810 DATA 134,144,15,192,141,176 ,11,173,68,3,197,254,208,4, 201,1,240,230 :REM*13	1040 DATA 220,41,16,73,16,133,2 ,173,1,220,41,16,73,16,5,2 ,133,2,96,0,0,0 :REM*3
590	DATA 62,3,230,21,224,238,20	820 DATA 202,138,166,252,157,0,	1050 DATA 0,0,0,0,0,0,0,0,0,0,2

# GAMES

52,0,1,254,0,3,135,0,3,0	1150 DATA 73,67,32,67,79,77,80,	14,15,0,0,1,3,1	:REM*182
,3,3,0,3,3,0,3,3	79,83,69,68,32,66,89,32,75	1250 DATA 3,1,3,1,3,1,3,1,3,	
DATA 0,3,135,0,1,254,0,0,2	,69,78,32,70,79	,1,3,4,4,4,8,6,7,7,4,4,4,8	
52,0,0,0,0,0,0,0,0,0,0	1160 DATA 85,78,84,65,73,78,147	,6,7,7,4,4,4,8,6	:REM*96
,0,0,0,0,0,0,0,0,0	,5,8,142,0,0,0,4,5,6,5,3,5	1260 DATA 7,7,4,0,4,0,4,1,3,1,3	
DATA 0,0,0,0,0,0,0,0,0,0	,6,5,4,5,6,7,8,4	,1,3,1,3,1,3,1,3,1,3,1,3,1	
,0,0,0,0,0,0,0,0,0,0	1170 DATA 5,6,5,4,5,6,5,4,5,6,7	,3,1,3,1,3,1,3,1	:REM*14
,0,0,252,0,0,252	,8,8,8,8,12,10,11,11,8,8,8	1270 DATA 3,1,3,1,3,1,1,8,2,2,2	
DATA 0,0,252,0,0,252,0,0,1	,12,10,11,11,8,8	,2,2,2,2,2,2,2,2,2,2,2,2	
20,0,0,0,0,0,0,0,0,0,0	1180 DATA 8,12,10,11,11,8,0,8,0	,136,130,1,2,1,1	:REM*122
,0,0,0,0,0,0,0,0,0	,8,4,5,6,5,4,5,6,5,4,5,6,7	1280 DATA 1,136,130,1,2,1,1,1,1	
DATA 0,0,0,0,0,0,0,0,0,176	,8,4,5,6,5,4,5,6	36,130,1,2,1,1,1,8,2,1,1,2	
,195,195,195,195,195,195,1	1190 DATA 5,4,5,6,7,8,4,5,6,5,4	,2,2,2,2,2,2,2,2	:REM*138
95,195,195,195	,5,6,5,4,5,6,7,8,4,5,6,5,4	1290 DATA 2,2,2,2,2,2,2,2,2,2	
DATA 195,195,195,195,174,1	,5,6,5,4,5,6,5,4	,2,2,2,2,2,2,2,2,2,2,4,0	
94,32,80,76,65,89,69,82,32	1200 DATA 1,8,1,1,1,1,1,1,1,1,1	,0,4,4,4,4,4,4,5	:REM*35
,35,49,58,32,32	,1,1,1,4,1,1,1,1,1,1,1,1,1	1300 DATA 4,4,5,4,4,4,4,4,4,3	
DATA 32,194,194,32,80,76,6	,1,1,1,4,136,130	,3,3,4,4,4,4,4,4,4,3,3,3,4	
5,89,69,82,32,35,50,58,32,	1210 DATA 129,2,1,1,1,136,130,1	,4,4,4,4,4,4,3,3	:REM*45
32,32,194,173	29,2,1,1,1,136,130,129,2,1	1310 DATA 3,4,4,4,4,3,4,4,5,4,4	
DATA 195,195,195,195,195,1	,1,1,8,4,1,1,2,1	,5,4,4,5,4,4,4,0,1,2,2,2,2	
95,195,195,195,195,195,195	1220 DATA 1,1,1,1,1,1,1,1,1,1,1	,8,4,4,8,4,4,2,2	:REM*233
,195,195,189,19	,4,1,1,1,1,1,1,1,1,1,1,1	1320 DATA 2,2,2,2,1,1,1,1,2,2,2	
DATA 29,84,65,71,45,74,79,	,4,1,1,1,1,1,1,1	,2,2,2,1,1,1,1,2,2,2,2,2	
72,78,32,70,69,68,79,82,46	1230 DATA 1,1,1,1,1,4,1,1,1,1,1	,1,1,1,1,2,2,2,2	:REM*177
,32,32,80,82,69	,1,1,1,1,1,1,1,4,0,0,71,15	1330 DATA 2,2,1,1,1,1,1,2,2,2,8	
DATA 83,83,32,66,85,84,84,	2,71,12,233,97	,8,4,4,8,4,4,8,4,4,8,2,2,4	
79,78,32,84,79,32,66,69,71	1240 DATA 104,143,48,143,24,210	,0	:REM*29
,73,78,77,85,83	,0,5,5,6,7,7,8,9,10,11,12,		

## JASON-RANHEIM CARTRIDGE MATERIALS FOR YOUR COMMODORE 64 or 128

*Quality Products  
from the World Leader!*

- Promenade C1 EPROM Programmer
- Game Type Cartridges
- Bank Switching Cartridges
- RAM/ROM Combination Cartridges
- Capture Archival Cartridge System
- Cases, EPROMS, Erasers, Etc.

*Call or write for complete information!*

Call Toll Free  
from California  
Tech Support

800-421-7731  
800-421-7748  
916-823-3284



**JASON-RANHEIM**  
1805 Industrial Drive  
Auburn, CA USA 95603

## FACTORY AUTHORIZED COMMODORE REPAIR CENTER 1-800-772-7289

(312) 879-2888 IL

C64 Repair (PCB ONLY) .	42.95	Amiga Repair	
C128 Repair (PCB ONLY) .	64.95	(PCB ONLY) . . . . .	99.95
1541 Permanent		Amiga Drive	
Alignment . . . . .	29.95	Repair . . . . .	149.95
1541 Repair . . . . .	79.95	Printers . . . . .	CALL
1571 Repair . . . . .	79.95	Monitors . . . . .	CALL
		Other Equipment . .	CALL

**CALL BEFORE SHIPPING  
PARTS AND LABOR INCLUDED  
FREE RETURN SHIPPING**

(APO, FPO, AIR ADD \$10.00)

**24-48 HR. TURNAROUND**

(Subject to Parts Availability)

**30 DAY WARRANTY ON ALL REPAIRS**

## COMMODORE PARTS

C-64 Power Supply . . . . .	34.95
128 Power Supply . . . . .	59.95
C-64 Over Voltage Sensor . . . . .	19.95
Other Parts . . . . .	CALL

(Plus \$3.00 Shipping/Handling)

All parts for Commodore equipment usually in stock

For Parts Call (312) 879-2350

Dealer Discounts Available

**TEKTONICS PLUS, INC.**

150 HOUSTON STREET  
BATAVIA, IL 60510

CLIP AND SAVE



MasterCard

VISA



# Gravitron



*Battle the pirate pilots by destroying their defense systems and fueling depots.*



By CHARLES ORCUTT

**S**tarbase Intelligence has discovered that pirate pilots from hostile spaceships are plotting to invade Earth to steal precious water. As an Earth-based pilot, you must attack the asteroids that provide shelter and sanctuary for the pirates. Once you eliminate the asteroids' automatic defense systems, your engineers can dismantle their fueling depots to keep the pirate ships from flying. That's the scenario for *Gravitron*, a smooth-scrolling arcade-style game for the C-64.

Listings 2 and 3 are Basic hex loaders that must be typed in, saved to disk and individually run to create machine language files. Listing 1 is the boot program that loads in and activates the files created by Listings 2 and 3.

Most of the game control is accomplished with a joystick in port 2. You can pause the action by pressing F7 and reset the game with F1. One caution: *Gravitron* is not compatible with all fast-load cartridges.

## THE ACTION

You move your ships by rotating them to face in the direction you wish to go and then applying thrusters. The forward thrusters are activated by pushing forward on the joystick, the vertical thrusters by pulling back on the joystick. To slow a ship down, rotate it 180 degrees and apply thrusters.

Asteroid defenses are destroyed by firing antimatter pellets at them with the fire-button on your joystick. You must be flying at a relatively low altitude for the pellets to work, and your ship will turn green to tell you when the altitude is appropriate.

The defensive machines on the asteroid surface are tricky. They don't move at the first two levels of play, but come to life later and seem to shift direction in a random manner. However, you'll

learn that it's not random at all. To evade their firepower, increase your altitude, but keep in mind that *your* firepower will become ineffective, as well.

Of course, your thrusters consume fuel, so, even while you're trying to destroy the enemy defenses, you must use their fuel depots. Your ship will turn green when its tanks are low enough to be filled. To obtain fuel, maneuver the ship so it hovers above a fuel depot, then let it fall low enough to make contact. Don't fall too hard, though, or you'll destroy the ship. The longer you're in contact with a fuel tank, the more fuel you'll receive, as indicated by the horizontal bar at the bottom of the screen. A low-frequency bell indicates fueling is in progress; a high-frequency bell sounds when the tank is full.

Once you eliminate three enemy defenses, your engineers can land on the asteroid and dismantle its fuel depots. You must then move on to another asteroid. Each successive one offers a higher level of play, where the gravity is stronger. Of course, fighting the additional gravity means consuming more fuel. The program also generates a new graphics layout for each level. At every fourth level you complete, you get an extra ship if you currently have fewer than three.

At certain levels, you'll confront an indestructible surprise. It won't harm your ship while you're refueling, but it's a killer otherwise. On other levels, you'll encounter a "black hole," which blinks slowly until you're within its range. Then it blinks quickly and pulls your ship toward it with double gravity. You can best pull away from a black hole's destructive force by selecting a vector at a multiple of 90 degrees. In other words, point your ship directly to the right, down, left or up, and apply thrusters.

When you've lost all your ships, the

game is over. Your score is calculated from the defenses you've destroyed, with defenses at higher levels yielding higher scores. The program saves the top ten scores in a disk file.

## PROGRAMMERS ONLY

*Gravitron* uses a technique known as smooth scrolling, which is done by changing the lower three bits of locations 53265 (vertical) and 53270 (horizontal). When an overflow value results from these changes, the main screen position pointer is updated in the appropriate direction. Then an algorithm copies the source data into the target screen area.

There are two different screen areas. At this point, the screen base location is changed, telling the VIC chip to look at a different area. I use \$7800 and \$7C00 as the screen locations. The screen is split at the bottom to allow the user to see various game variables, such as fuel, ships remaining, and so on. I use \$7000 as the location for the background graphics, but the lower portion uses \$6800 for the character set. This is needed to achieve a smooth transition between the game scene and the status indicators.

The indicators are static with respect to the scrolling game by means of a dual raster interrupt. This interrupt also changes screen and border colors, VIC character location and sprite color and priority.

There is a pseudo-random screen generator that operates every time you get to a new level. It selects from the graphics the imagery it will paste to the screen map zone. The map zone is layered in nine 1K blocks of memory and is configured to allow for vertical wrapping without any sign of flicker. That simply means that the top three screens are a replica of the bottom three. The

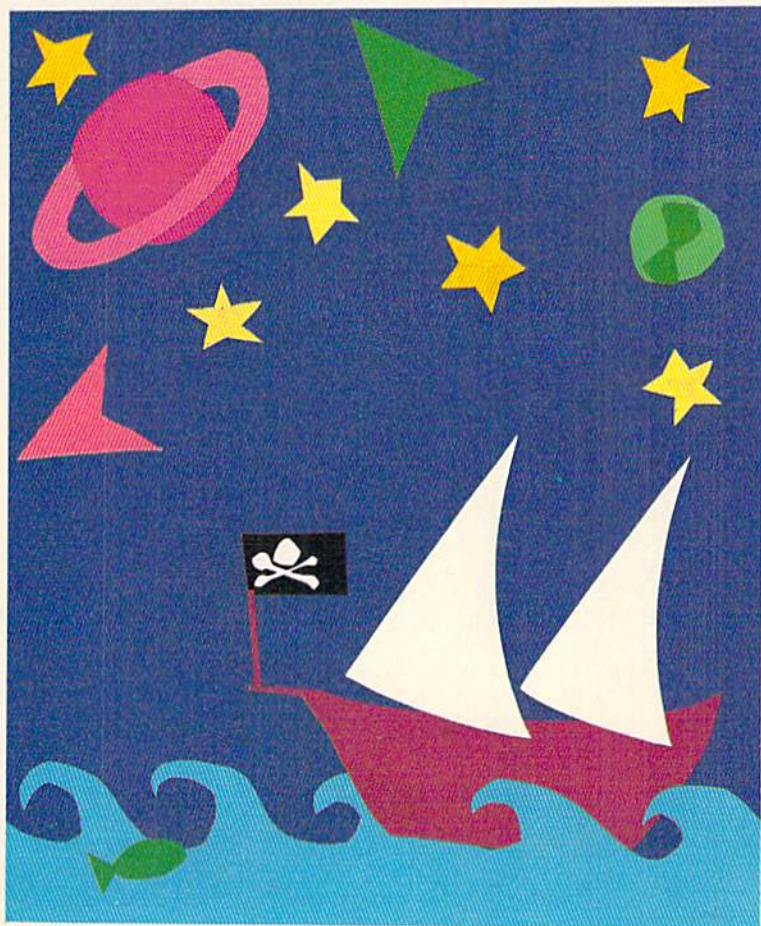
RUN it right: C-64; joystick

generator uses a scheme to verify that repetition is minimized in the resulting map. It operates quickly enough to be invisible to the user, yet does a remarkable piece of art work each time it executes.

To destroy a sprite, I use a technique that has no name I'm aware of. It might well be called disintegration. It works by ANDing the sprite definition with numbers that are derived by a random generator. This was a memory conservation effort, but it results in a realistic destruction effect.

The ships (there are sixteen images for these) require only 32 bytes each to define, instead of the usual 64. This crunched the program by 512 bytes. The pellet sprite was made by clearing its zone and STAing three numbers in place. This saved memory, too. The whole program was made to be as memory-efficient as possible. ☐

*Charles Orcutt is an electronics technician and a self-taught Basic and machine language programmer. Gravitron's graphics were created by Dan Diamante.*



#### Listing 1. Loader program.

```

5 IFA=0THENFORX=0TO13:READA:POK 40 PRINT"{10 CRSR DNs}":REM*189 70 A=3:LOAD"+GRAV CHARS",8,1
EX+49152,A:NEXT :REM*120 45 PRINT"{9 SPACES}"; :REM*140 :REM*129
10 IFA=0THENA=2:GOTO30 :REM*227 50 PRINT"LOADING "; :REM*102 75 SYS49152 :REM*241
15 IFA=2THEN65 :REM*253 55 PRINT"{CTRL 1}....."; :REM*147 80 A=4 :REM*147
20 IFA=3THEN75 :REM*9 85 SYS32768 :REM*44
25 IFA=4THEN85 :REM*24 60 A=2:LOAD"+GRAV ML",8,1 90 DATA169,0,160,0,153,253,118,
30 PRINTCHR$(14) :REM*126 :REM*99 153,0,119,200,208,247,96
35 POK53280,2:POKE53281,1:PRIN 65 PRINT"{CTRL 1}....."; :REM*157
T"(SHFT CLR)":REM*235 :REM*160

```

#### Listing 2. Gravitron program.

```

0 REM CREATE GRAVITRON ML E8:END :REM*78 25 FOR I=1 TO 30 :REM*181
:REM*132 15 IF LEN(A$)<62 THEN 55 30 C$=MID$(B$(I*2)-1,2):H$=LEF
5 OPEN 8,8,8,"+GRAV ML,P,W" :REM*254 T$(C$,1):L$=RIGHT$(C$,1)
:REM*188 20 B$=MID$(A$,1,20)+MID$(A$,22, 35 H=VAL(H$):IF H$>"9" THEN H=A
10 READ A$:IF A$="-1" THEN CLOS 20)+MID$(A$,43,20) :REM*242

```





# G A M E S

161	DATA 900620CD8C4C3C8738AD 0 3D0C929B00620CD8C4C 3C87AD1 0D02902F00EAD :REM*237	184	DATA 900FA9878D01D0A98E8D 0 9D0A9018D679960CE8C 99D0FAA D8A998D8C99EE :REM*106	207	DATA DCA9008D15D0A9978D00 D DA9158D18D0A91B8D11 D0A9C88 D16D060AD80099 :REM*239
162	DATA 02D038C945901120CD8C 4 C3C87AD02D038C90FB0 0320CD8 CCE8399D049A9 :REM*113	185	DATA 09D0CE01D0AD01D038C9 7 3B00FA9738D01D0A9A2 8D09D0A 9018DBE9360A9 :REM*43	208	DATA C918F007AD8099C917D0 1 738AD8199E9708D8199 AD8099E 9008D809938E9 :REM*70
163	DATA 078D8399AD6699F00DAD 0 0DCA82904F021982908 F0034C8 A87EEF87BEEF8 :REM*11	186	DATA 3F8D02DDA9C68D00DDA9 E D8D18D0A9D08D16D0A9 138D11D 060A00020118C :REM*136	209	DATA 178D809960AD80993005 A D8099D01718AD819969 708D819 9AD809969008D :REM*13
164	DATA 7FADF87B38C9119021A9 0 18DF87F8DF87B4C8A87 CEF87FC EF87BADF87B38 :REM*75	187	DATA A200A0002000C018A5FB 6 92885FBA90065FC85FC 18A5FD6 97885FDA90065 :REM*155	210	DATA 00991869178D809960AD 1 0D029FD8D10D0AD15D0 29FD8D1 5D0A9008D7699 :REM*125
165	DATA C901B008A9108DF87F8D F 87BAD6699F02FAD00DC 2901D02 820548DACF87B :REM*107	188	DATA FE85FEE8E014D0DC60AD 7 D09D06318AD0ED06901 8D0ED09 008AD10D04980 :REM*15	211	DATA 60AD10D029F78D10D0AD 1 5D029F78D15D0A9008D 789960A 9018D6699A9E4 :REM*40
166	DATA 8CF99CE8D99D009B961 9 88D8D99200588CE8E99 D00CACF 87BB973988D8E :REM*65	189	DATA 8D10D018AD0CD069018D 0 CD09008AD10D049408D 10D018A D0AD069018D0A :REM*27	212	DATA 8D987B8D987FA000A2E0 8 A99997B99997FE8C8C0 05D0F3A 005A9E899987B :REM*233
167	DATA 99209888206E88200189 6 038AD06D0F9CD988D06 D0B008A D10D049088D10 :REM*136	190	DATA D09008AD10D049208D10 D 018AD04D069018D04D0 9008AD1 0D049048D10D0 :REM*35	213	DATA 99987FC8C01ED0F5A9E4 9 9987B99987FC8C028D0 F5A0279 9707B99707F88 :REM*37
168	DATA D060AD06D01879CD988D 0 6D09008AD10D049088D 10D0603 8AD07D0F9DF98 :REM*151	191	DATA EE7F99AD7F99C908D00D A 9008D7F99A9038D7C99 8D7D996 0AD7D99D05E38 :REM*185	214	DATA 10F7A0188C7499ACBF93 F 00BA9F099B77B99B77F 88D0F76 0CE6F99D025A9 :REM*196
169	DATA 8D07D06018AD07D079DF 9 88D07D060ACF87BB985 98A8290 8D006982904D0 :REM*61	192	DATA AD0ED0E9018D0ED0B008 A D10D049808D10D038AD 0CD0E90 18D0CD0B008AD :REM*87	215	DATA 08AD6F99AC7499D006A9 0 08D66996038B99D7BE9 01C9E4D 005CE7499A9E9 :REM*31
170	DATA 2D60AD8499C901F010CE 8 599D005A9018D8599A9 028D849 960EE8599AD85 :REM*168	193	DATA 10D049408D10D038AD0A D 0E9018D0AD0B008AD10 D049208 D10D038AD04D0 :REM*110	216	DATA 999D7B999D7F60A9018D 6 699AC7499C0818F023A9 208D07D 4A94E8008D4D0 :REM*82
171	DATA 99C90AD00AA9098D8599 A 9008D849960AD8499C9 02F010C E8599D005A901 :REM*124	194	DATA E9018D04D0B008AD10D0 4 9048D10D0CE7F99100D A9078D7 F99A9018D7C99 :REM*217	217	DATA 9D7B186901C9E9D008EE 7 499AC7499A9E5999D7B 999D7F6 0A9308D07D4A9 :REM*253
172	DATA 8D8599A9018D849960EE 8 599AD8599C90AD00AA9 098D859 9A9008D849960 :REM*37	195	DATA 8D7D9960AD7D99D023EE 0 FD0EE0DD0EE0BD0EE05 D0EE7E9 9AD7E99C908D0 :REM*141	218	DATA 758D08D46020F28DA9D F 8 D18D4A9108D00D4A927 8D01D4A 9228D05D4A921 :REM*148
173	DATA AD8499C902F005C901F0 1 060CE8699D009AD8599 8D86992 0A28A60CE8699 :REM*87	196	DATA 0DA9008D7E99A9048D7C 9 98D7D9960AD7D99D01E CE0FD0C E0DD0CE0BD0CE :REM*114	219	DATA 8D06D4A9118D0CD48D0D D 4A9188D13D4A9888D14 D4A9A88 D0ED4A9618D0F :REM*218
174	DATA D009AD85998D86992039 8 A60ACF87BB99798A829 01D0069 82902D02D60AD :REM*35	197	DATA 05D0CE7E99100DA9078D 7 E99A9028D7C998D7D99 60AD19D 08D19D02901D0 :REM*97	220	DATA D460A018A9009900D488 1 0FA60A000A2EC8A994A 7B994A7 FC8E8E0F1D0F3 :REM*241
175	DATA 8799C901F010CE8999D0 0 5A9018D8999A9028D87 9960EE8 999AD8999C90A :REM*96	198	DATA 034C31EAAD12D0C9FDB0 4 2A0FD8C12D0AD18D029 F0090A8 D18D0AD11D029 :REM*98	221	DATA A000A2EA8A99577B9957 7 FC8E8E0F1D0F3604A4A 4A4A602 90F1869F19950 :REM*46
176	DATA D00AA9098D8999A9008D 8 79960AD8799C902F010 CE8999D 005A9018D8999 :REM*98	199	DATA F81869078D11D0AD16D0 2 9F81869038D16D0A900 8D21D0A 0079927D088D0 :REM*204	222	DATA 7B99507F60F8ADBA9338 C DC293902CD014ADB993 38CDC19 39021D009ADB8 :REM*199
177	DATA A9018D879960EE8999AD 8 999C90AD00AA9098D89 99A9008 D879960AD7A99 :REM*105	200	DATA FA8D25D0A9FF8D1BD0A9 0 58D23D04CF18BAD11D0 1829F86 D7E998D11D018 :REM*96	223	DATA 9338CDC0939016ADB993 8 DC193ADB8938DC093AD BA938DC 293D820C58ED8 :REM*41
178	DATA 49018D7A99AD8799C902 F 005C901F01560CE8B99 D00EAD8 9998D8B99AD7A :REM*92	201	DATA AD16D029F86D7F998D16 D 0AD18D029F0090C8D18 D0A0D08 C12D0AD03808D :REM*8	224	DATA 60F818AD8B936DBB938D B 8939014ADB993186901 8DB9939 00918ADBA9369 :REM*185
179	DATA 99F00320068B60CE8B99 D 0FAAD89998D8B99AD7A 99F0032 02F8B60AD8899 :REM*57	202	DATA 21D0A007B9D999927D0 8 8D0F7A90F8D25D0A900 8D1BD0A D07808D23D068 :REM*99	225	DATA 018DBA93D8ADB8932020 8 EA00420258EADB893A0 0520258 EADB99320208E :REM*62
180	DATA C901F010CE8A99D005A9 0 18D8A99A9028D889960 EE8A99A D8A99C90AD00A :REM*184	203	DATA A868AA6840AD16D029F8 1 86D7F998D16D060AD11 D029F81 86D7E998D11D0 :REM*104	226	DATA A00220258EADB993A003 2 0258EADBA9320208EA0 0020258 EADBA93A00120 :REM*78
181	DATA A9098D8A99A9008D8899 6 0AD8899C902F010CE8A 99D005A 9018D8A99A901 :REM*168	204	DATA 60A9EF85FDA91885FEA9 0 085FBA978186D7B9985 FC18A5F D6D829985FDA5 :REM*141	227	DATA 258E60ADC09320208EA0 1 320258EADC093A01420 258EADC 19320208EA011 :REM*83
182	DATA 8D889960EE8A99AD8A99 C 90AD00AA9098D8A99A9 008D889 960AD8899C902 :REM*141	205	DATA FE690085FE18A5FD6D81 9 985FDA5FE690085FE18 A5FE6D8 09985FE60A980 :REM*112	228	DATA 20258EADC193A0122025 8 EADC29320208EA00F20 258EADC 293A01020258E :REM*153
183	DATA F005C901F02A60CE8C99 D 023AD8A998D8C99CE09 D0EE01D 0AD01D038C987 :REM*56	206	DATA 8D1AD078A9318D1403A9 E A8D150358A9008D20D0 8D21D0A D0EDC09018D0E :REM*57	229	DATA 60A000B908991869E099 6 67B99667FC8C005D0EF 60AD569 9D00AADBF93C9 :REM*205

# GAMES

230	DATA 03F003EEBF93201B90EE B C93ADBC9308AD5999F8 18ADBB9 369018DBB93D8 :REM*32	253	DATA 9918A5FB690185FBA5FC 6 90085FC4CA791AD6399 85FBAD6 49985FC601869 :REM*224	276	DATA C0CF0CF0C3CF0CF0CF0C FCFCFCFCFCFCFCFCFCFCFCFCFC F808080808080808 :REM*38
231	DATA ADBB93290FA01D20258E A DBB9320208EA01C2025 8E60A9F F8D0FD4A9808D :REM*251	254	DATA 018D5F9918A5FB697885 F BA5FC690085FCAC6099 88202E9 28D6199AC6099 :REM*29	277	DATA 08080A0A0A0A0A0A0A0A0A 8A8A8A8A8A8A8A8A8A8A8A8A AAAAA000000000 :REM*226
232	DATA 12D4AD1BD460A98085A3 A 94485A420D48F4C8A8F 20D48FA 90085A3A94085 :REM*113	255	DATA 202E9238ED61998D6199 3 8A920ED61998D619938 A5FBED6 19985FBA5FCE9 :REM*30	278	DATA 0000000000333333333333 3333F0C0C0C0C0C0C0C3F 0F30303 00F03033C0F30 :REM*118
233	DATA A4ACF87B18A5A3694085 A 3A5A4690085A488D0F0 20CD8C2 0E38CA03FA200 :REM*79	256	DATA 0085FC4CA791A9508DA7 9 2A9928DA892AE5D99BD A992186 DA7928DA792AD :REM*108	279	DATA 3030303030300F0C333333 3 333330C3C333333333F3C 333333F3 0330303030303F :REM*242
234	DATA 8E20D0E8D0FA20508F31 A 391A38810FA003FA200 8E20D0E 8D0FAB1A3D006 :REM*16	257	DATA A89269008DA8926CA792 B 9CA9260B9DC9260B9EC 9260B9F F9260B90B9360 :REM*53	280	DATA 2A2222222222222A0828 0 8080808082A2A020202 2A20202 A2A0202022A02 :REM*212
235	DATA 810F34CBB8F4C8A8F60 A 00FB9A8939900D08810 F7208C9 28D04D0208C92 :REM*128	258	DATA B9379360B9469360B95D 9 360B9969360208C92AC 5B99B90 010CD5D99F0F2 :REM*178	281	DATA 022A222222222A020202 2 A20202020A02022A2A2 20202A2 222A2A020202 :REM*239
236	DATA 8D05D060A9028D01D48D 0 8D48D0FD4A9228D05D4 8D06D48 D0CD48D0DD48D :REM*102	259	DATA AC5B99B900108D5D9960 A 9FF8D0FD4A9808D12D4 AD1BD48 D5B9960A9EF85 :REM*30	282	DATA 020202022A2222222A22 2 22A2A222222A020202 2 3030303 0303030303333333 :REM*176
237	DATA 13D48D14D4A9818D04D4 8 D0BD48D12D460A000B9 1599C9F FF007995399C8 :REM*120	260	DATA FBA91885FC6003000000 0 4080C1014181C20090B 080C020 C09060A0F0A0E :REM*145	283	DATA 333333333330CFF0FBFBF A BFF000000000000000 00180000 01800001BD8001 :REM*158
238	DATA 4C019060A000A264C8D0 F DE8D0FA60A9008D18D4 204A8CA 99320D2FFA908 :REM*187	261	DATA 0F90F09010012000113 1 4150701FF0001020304 0506072 12223242526FF :REM*166	284	DATA FF800199800018000000 0 000000000000000000 00000000 C0003180007B6 :REM*5
239	DATA 8D5C99A000AD5C999900 1 0CE5C991005A9088D5C 99C8D0E DA9EF85FBA918 :REM*2	262	DATA 0A01090A0B0C0D0E0F10 1 11213142E0F07011516 1718191 A1B1C1D343536 :REM*156	285	DATA 0007FE00006CE00000C00 0 000000000000000000 00000000 00000000300E0 :REM*31
240	DATA 85FCA9FFA000A22391FB C 8D0FBE6FCCA10F6EE5E 99AD5E9 9C912D023A9EF :REM*151	263	DATA 3738393AFF0501404142 4 34445466263FF140146 4748494 A4B4C4D4E4F50 :REM*14	286	DATA 0703C007FE0000BC0003 3 800000000000000000 00000000 0000000780000F :REM*198
241	DATA 85FBA91885FCA95F85FD A 93085FEA000A20BB1FB 91FDC8D 0F9E6FCE6FECA :REM*121	264	DATA 5125354555657585965 6 66768696A6B6C6D6E6F 7071727 3747576777879 :REM*58	287	DATA 81E00FFF0003F800003F0 0 000000000000000000 00000000 000000000780 :REM*215
242	DATA 10F260208D904C5A90AC 5 E99A98885FBA91785FC E6FC18A 5FB696885FBA5 :REM*67	265	DATA FF05013A3B3C3D3E3F5A 5 B5C5D5E5FFF06027F80 8182838 485A0A1A2A3A4 :REM*135	288	DATA 0007E00007FFE007FFF0 0 7E0000000000000000 00000000 0000000018000 :REM*78
243	DATA FC690085FC88D0EEA900 8 D629920889120E49018 A5FB690 185FBA5FC6900 :REM*216	266	DATA A5C0C1C2C3C4C5C6FF11 0 2858788898A8B8C8D8E 8F90919 29394959697A7 :REM*166	289	DATA 01E00007E00007FE0007 F FC0000070000000000 00000000 0000000300000 :REM*61
244	DATA 85FCA00120E292A838A5 F BE97885FBA5FCE90085 FC88D0F 0AD629938C978 :REM*137	267	DATA A8A9AAABACADAEAFB0B1 B 2B3B4B5B6B7B8C7C8C9 CACBCCC DCECFD0D1D2D3 :REM*102	290	DATA 380006300007F00003FE 0 000007800001E000000 00000000 00000000000 :REM*72
245	DATA B0034CAF9060A002202E 9 28D5F99A0028C6099EE 6099AC6 099202E92C9FF :REM*136	268	DATA D4D5D6D7D8FF0601B8B9 B ABBBBCDBEBFD9DADBDC DDEDFF FAE73AE730000 :REM*124	291	DATA 0000010003230003FF00 0 33800001C00000E0000 00000000 00000000000 :REM*249
246	DATA F02B38E901CD5F99D03E 1 869018D5F99AAA000AD 6599C9F FD0038A91FB18 :REM*144	269	DATA 0000AEA214460000AC8 0 0000000000032030000 00020A0 8070D05030E06 :REM*59	292	DATA 00000000180001998001 F F8001BD80001800018 00000000 00000000000 :REM*167
247	DATA A5FB690185FBA5FC6900 8 5FC4CF190A000202E92 AA186D6 2998D62998A4A :REM*201	270	DATA 0A0409020A0807605255 4 E604D4147415A494E45 6050524 F55444C596050 :REM*80	293	DATA 00000000000800000C4 C 000FFC00001CC0003800 0070000 00000000000 :REM*245
248	DATA 1865FB85FBA5FC690085 F C601869018D5F9918A5 FB69788 5FBA5FC690085 :REM*166	271	DATA 524553454E5453606E6E 6 E6E6E60475241564954 524F4E6 0425960434841 :REM*62	294	DATA 00000000000C00001C00 0 00C60000FE0007FC001 E000078 00000000000 :REM*231
249	DATA FCAC609988202E928D61 9 9AC6099202E9238ED61 998D619 938A920ED6199 :REM*39	272	DATA 524C4553604F52435554 5 4605749544860475241 5048494 3536042596044 :REM*32	295	DATA 00000000000000000000 0 1800007800007E0007F E003FFE 00E00000000 :REM*186
250	DATA 8D619938A5FBED619985 F BA5FCE90085FC4CF190 207492A 002202E928D5F :REM*120	273	DATA 414E604449414D414E54 4 56E60434F5059524947 4854607 1797878604849 :REM*68	296	DATA 00000000000000000000 0 00000000001E00007E0 07FFE00 FFFE00007E000 :REM*238
251	DATA 99A0028C6099A9FF8D65 9 9A5FB8D6399A5FC8D64 99EE609 9AC6099202E92 :REM*18	274	DATA 54605350414345604F52 6 0464952456042555454 4F4E605 04F5254607260 :REM*93	297	DATA 00000000000000000000 0 000000000001E007 81F000F FF0001FC0000F :REM*33
252	DATA C9FFF02B38E901CD5F99 D 02E1869018D5F99A000 B1FBC9F FF005A9008D65 :REM*200	275	DATA 544F60504C4159606E6E 6 E6E6E6FFC0CF0C3CF0C CFCFCFC CCCCCCCCCF3 :REM*226	298	DATA C0000000000000000000 0 000000000000000000 00070C 003C0E0007FE0 :REM*8

# GAMES

```

299 DATA 003D00001CC000000000 0 D8DA39DA90B8DA49DA9 028D20D 3A5A4E90085A4 :REM*8
000000000000000000000000 00300000 0A9008D21D0A9 :REM*181 340 DATA 38A5FDE90685FDA5FE9 0
018C0006DE0000 :REM*155 320 DATA 9320D2FF20E7FF203D9C 2 085FE60A002B1FD91FB 8810F96
300 DATA 7FE0000736000300000000 0 0339CA90020D5FF20B7 FFC940F 020E7FF203D9C :REM*129
000000000000000000000000 00000000 00320579CA993 :REM*195 341 DATA 20339CA90085A5A96085 A
00000000000000 :REM*9 321 DATA 20D2FFA901A000990D8 9 6A23DA060A9A520D8FF 60A90BA
301 DATA 000000000000000000000000 0 900D99900DA99E8DAC8 D0F1A00 238A09E20BDDF :REM*140
000AA000296800A55A0 3E96BCC 0B9AC9DF00720 :REM*245 342 DATA 60A900A208A0FF20BAFF 6
FAAF3FFEBFFFF :REM*52 322 DATA D2FFC84CDE9920C79BA0 0 018290F693091FB60AD 00DC291
302 DATA FFFFFFFF3FFF00FF F 3B1FD8DA59DC8B1FD8D A69DC8B 0D0F96020719C :REM*216
0003C00000000000000000 00000000 1FD8DA79D2043 :REM*176 343 DATA A200A0000BD9D990060 E
000200002AA00 :REM*134 323 DATA 9BCEA49DF00BADA39DD0 1 8C8E006D002A200C03C D0EE60A
303 DATA 2AAA0802AA80C20C030 3 B20D89B4CED99A000B9 089EF00 901A208A00F20 :REM*147
0301555507FFF45555 547FFD 720D2FFC84C15 :REM*166 344 DATA BAFFA90020BDDF20C0FF A
D7F57DD7F7FDD :REM*107 324 DATA 9A206B9B204F9C60A5FD 8 20120C6FF20CFFF2490 50F920C
304 DATA 7F5FDD7F7FDD7F7FDD15 5 DA89DA5FE8DA99DCEA4 9DF026A CFFA90120C3FF :REM*216
5543FFFC3FFFC3FFF FC000000 93085FDA96085 :REM*35 345 DATA 60AD10D02904D00C38AD 0
00300000000000 :REM*237 325 DATA FEA93685A3A96085A4A0 0 0D0CD04D090034CD59C 38AD01D
305 DATA 000000000000000000000000 0 5B1FD91A38810F9CEA4 9DF008A 0CD05D09010A9 :REM*58
5E02D55782D55 :REM*60 326 DATA 9AADA89D85FDADA99D85 F 0D0CD04D090034CD59C 38AD01D
306 DATA 780B55E002D78000BE00 0 EA002A90191FD8810FB A003ADB A99A90D60A9008D5799 A9018D5
0280000000000000000000 00000000  A9391FDC8ADB9 :REM*149 899A90C8D5A99 :REM*100
00000000008400 :REM*187 327 DATA 9391FDC8ADB89391FD20 6 347 DATA A9116038AD01D0CD05D0 9
307 DATA 000000000000000000000000 0 B9BA000B9D39DF00720 D2FFC80 012A9018D5799A9008D 5899A90
00000000AA80020020 083F082 C889AADA89D85 :REM*80 48D5A99A90960 :REM*246
0C0C223043223 :REM*154 328 DATA FDADA99D85FEE8D0FDC8 C 348 DATA A9018D57998D5899A900 8
308 DATA 113223113223043220C0 C 080D0F8206B9BACAA9D ADA89D8 D5A99A9056018AD0CD0 AC77997
2083F0802002000AA80 00000000 5FDADA99D85FE :REM*246 9CD988D0CD000 :REM*42
00000000000000 :REM*112 329 DATA B1FD498091FDAD00DC29 1 349 DATA 08AD10D049408D10D018 A
309 DATA 0000740404040320102 0 0F063AD00DCC97EF00F C97DF01 DDD079DF988D0DD060 CE5399F
3040404030201020304 0401010  DC97BF029C977 :REM*239 00160A9038D53 :REM*245
2030404040302 :REM*176 330 DATA F0384C9E9A18B1FD6901 2 350 DATA 99AD10D0298D01DAD0E D
310 DATA 01020304040403020180 8 97FC91BD002A90191FD 4C9E9A3 038C9E0B01538C97C90 1038AD0
00808080808080808080804 0404040 8B1FDE901297F :REM*189  FD0C9AFB00838 :REM*223
4040480800101 :REM*2 331 DATA D002A91A91FD4C9E9AB1 F 351 DATA C94B900320529D60CEBE 9
311 DATA 01018002020202020202 8 D297F91FDCEAA9D1005 A9008DA 3CEBE93CEBE93AD1499 49018D1
0010101010000010101 0101010  A9D4C9E9AB1FD :REM*230 49938AD00D0CD :REM*185
1000202020202 :REM*194 332 DATA 297F91FDEEAA9DADAA9D C 352 DATA 0ED0900334C869D38AD01 D
312 DATA 02020000020202020001 0 903D005A9028DAA9D4C 9E9A9A C 0CD0FD0900720168820 A988602
1010101010100020202 0200000  A9DADA89D85FD :REM*69 0168820D58860 :REM*146
2030405040302 :REM*226 333 DATA ADA99D85FEB1FD297F91 F 353 DATA 38AD01D0CD0FD0900720 4
313 DATA 00020304050403020000 0 D204F9C206B9B20189C 60F8ADB 28820A9886020428820 D588600
3030201000102030303 0201000  A9338CDA59D90 :REM*28 1010100000000 :REM*141
1020303190B0D :REM*54 334 DATA 1CD014ADB99338CDA69D 9 354 DATA 0000000000000000058E 2
314 DATA 1115191614110F111416 1 011D009ADB89338CDA7 9D9006A 202020494E055542059 4F55522
915110D0BB1FD91FBC8 1B101C1 9018DA39DD8D8 :REM*218 0494E49544941 :REM*167
01B0101050100 :REM*138 335 DATA 6020C79BA209200E9C20 D 355 DATA 4C532057495448204A4F 5
315 DATA 06020003050000000000 0 89BCA10F720C79BA209 A003B1F 9535449434E000D0D0D 0D0D0D0
4000800000000000000000 0000010 D8DAB9D20208E :REM*235  D0D0D0D0D0D0D :REM*251
0000300070C05 :REM*194 336 DATA C820479CADAB9DC82047 9 356 DATA 0D0D0D0D0D0D0D0D20202 2
316 DATA 040A0000000019000000 0 CA004B1FD8DAB9DC8C8 20208E2 02020594F5520484156 45204D4
040200040904 :REM*207 0479CADAB9DC8 :REM*41 1444520544845 :REM*4
317 DATA 09040401FF0305000000 0 337 DATA 20479CA005B1FD8DAB9D 2 357 DATA 20544F502054454E2100 0
00004000800000000000 0000000 0208EC8C820479CAD AB9DC82 D0D0D0D0D0D0D0D0D 0D0D0D0
0010000030007 :REM*203 0479C20D89BCA :REM*170  D0D0D0D0D0D :REM*120
318 DATA 0C05000A000000001900 0 338 DATA 10B860A92785FBA90585 F 358 DATA 2020202020202020202 2
008001901000000000404 1770000  CA90085FDA96085FE60 18A5FB6 020204E4F54204F4E20 544F502
7020404020004 :REM*31 92885FBA90065 :REM*30 054454E004030 :REM*87
319 DATA 090409040401A9008DAA 9 339 DATA FC85FC18A5FD690685FD A 359 DATA 3A544F5054454E4853
90065FE85FE6038A5A3 E90685A :REM*24

```

### Listing 3. Character Generator program.

```

0 REM CREATE GRAVITRON CHARACTE 15 IF LEN(A$)<62 THEN 55          T$(CS,1):L$=RIGHT$(C$,1)
RS :REM*236 :REM*254 :REM*209
5 OPEN 8,8,8,"+GRAV CHARS,P,W" 20 B$=MID$(A$,1,20)+MID$(A$,22, 35 H=VAL(H$):IF H$>"9" THEN H=A
:REM*28 :REM*242 SC(H$)-55 :REM*85
10 READ A$:IF A$="-1" THEN CLOS 25 FOR I=1 TO 30 :REM*181 40 L=VAL(L$):IF L$>"9" THEN L=A
E8:END :REM*78 30 C$=MID$(B$, (I*2)-1,2):H$=LEF SC(L$)-55 :REM*136

```

# GAMES

```

45 BY=H*16+L:PRINT#8,CHR$(BY);      116 DATA F0701A55000000000001 A 139 DATA 00000000A0FD00000000 0
      :REM*67                          658000000007070701C 04000000 000FE5500000000000 8075000
50 NEXT:GOTO 10                        :REM*115                          0000000000000000 :REM*247      0000000000054 :REM*133
55 IF LEN(A$)<21 THEN B$=A$:GOT      117 DATA 0A55000000000000AA55 140 DATA 00000000000000140000 0
      O 70 :REM*184                    0000000000AA9570000 0000000A 000000001500000000 0000000
60 IF LEN(A$)<42 THEN B$=LEFT$(      B57F1000000000 :REM*113          60000000000000 :REM*203
      A$,20)+RIGHT$(A$, (LEN(A$)-21 118 DATA 0000FC570001515010505 0 141 DATA 000220000000000000BC 0
      )):GOTO 70 :REM*176             100006A7FF0FFAA5500 AAFCC000 0000000000000000 0000000
65 B$=LEFT$(A$,20)+MID$(A$,22,2     005816001ABFFF :REM*117          00000000000000 :REM*23
      0)+RIGHT$(A$,LEN(A$)-42)      119 DATA 0005504102A9AA001550 0 142 DATA 00000000000000000000 0
      :REM*140                          2AA5500FF0A4B0AAB55 0000000C 0000000000000000 0000000
70 FOR I=1 TO LEN(B$)/2:REM*221      0C0C0C00000000 :REM*165          00000000000000 :REM*18
75 C$=MID$(B$, (I*2)-1,2):H$=LEF      120 DATA 00000000000002A90000 0 143 DATA 00000000000000000000 0
      T$(C$,1):L$=RIGHT$(C$,1)      0000209A4500000000A A550000 0000000000000000 0000055
      :REM*140                          000000000AA55 :REM*181          400000001061B :REM*68
80 H=VAL(H$):IF H$>"9" THEN H=A      121 DATA 075F0000000055FFFE00 0 144 DATA 505061B6CB0C1C05055 A
      SC(H$)-55 :REM*56                000000055FF80000000 0000A8D 000055005510410410 1040005
85 L=VAL(L$):IF L$>"9" THEN L=A      60100000000000 :REM*144          4024111111004 :REM*120
      SC(L$)-55 :REM*84                122 DATA 02A9540000000000806A 1 145 DATA 0401B0B0BCAC6C5F0602 0
90 BY=H*16+L:PRINT#8,CHR$(BY);      4000000000AA55000 00000000 00000000 000000005A01 0000000
      :REM*148                          0AA55150000000 :REM*81          0806AA55400000 :REM*43
95 NEXT:GOTO 10                        :REM*160                          123 DATA 0000FFAA550000000000 C 146 DATA 00002896540000000000 0
100 REM CHARACTER HEX DATA          0BFAA1500000000000C0 A05C000 0000501010100000000 FFF5F07
      :REM*67                          00000000000000 :REM*220          F1F050129D755 :REM*58
101 DATA 00700000AA55002A1500 0 124 DATA 000000000000003A90000 0 147 DATA 00F8DA450000F5550000 8
      0806A15259441010085 DF3C3F5 0000000FF550000000 0000FC5 0514015550000000095 6B58400
      555440055FA00 :REM*187          70000000000000 :REM*5          0000005451C00 :REM*192
102 DATA 0157D7C100409F2A6F6F 8 125 DATA 00FC0924900002090400 0 148 DATA 5F0701175F550000F13C F
      FCB00000FC6B0500F0 00000000 00000000000000101 09A4500 00055A9681BC57B1F1F 1F5F7FF
      0C07F6A0000000 :REM*74          00000FFA07C15 :REM*74          F5F87E1B8E0B0 :REM*119
103 DATA 000000C0A00000000000 0 126 DATA 000101000000005570C0 C 149 DATA E0F855FD5F0707071757 5
      0000000000000000100 00000000 0000707546661410100 00000000 5C0000C0C030C0FF6C6F 6A5B1B1
      000A05A0500000 :REM*148          00000000000000 :REM*244          A150000000C0C0 :REM*73
104 DATA 000202924904000012A55 0 127 DATA 000000000000005000000 0 150 DATA FCAB91B0000000000000 0
      0020A259040A8551F7A 7F55000 000000054000000000 00000000 00000010500000000000 00AA550
      00055FE0DA65 :REM*75            00000000000000 :REM*219          00000000500A9 :REM*255
105 DATA 00600A5400000AA550000 A 128 DATA 0100000000000005500 0 151 DATA 501010717F7F0005FDEA E
      B1F1C1C70601F05FF55 701C1F0 0000000000000054000000 00000000 59090E5005555AA5500 0040004
      58040F07F551C :REM*247          00000000000000 :REM*48          0E07A9F651A06 :REM*48
106 DATA F755000000FF5507FD55 0 129 DATA 05010101010000005554 7 152 DATA 000000A0FC5490001B1B 0
      0000000F850F0500000 00000000 00000000000000 00000000 00000000 601000000000C0FFABAA 5500000
      00000000000000 :REM*248          1050000AA5641 :REM*77          000FAD55540000 :REM*214
107 DATA 01550000000000005555 0 130 DATA 4040000000000A06A1500 0 153 DATA 00000181605A15050000 0
      71F1F07000AA55F6FD 00C00000 00000000002A956060601 0000AA5 5500000A55550000004106 1B6F7C5
      0AA55AA550000 :REM*247          5000080400A25 :REM*250          000000B0FC0000 :REM*212
108 DATA 000A954000C070700000 F 131 DATA 904000000000FF556C6C 1 154 DATA 00000000150000000000 0
      C5701000000000000000 FF55000 8070100C07F15000FD 540004F 00041000000000000000 8269140
      00000000000000 :REM*202          45400000000000 :REM*234          0000000000AA55 :REM*227
109 DATA 00000000000000000000 0 132 DATA 0605000000000000AA55 0 155 DATA 00000000000000000094905000 0
      00000000000000000000 00000000 0000000000054000000 00000000 00000000000000000000 00000000
      00000040000000 :REM*72          00000000000000 :REM*98          00000000000000 :REM*71
110 DATA 00000000410000000000 0 133 DATA 00000100000000000000 5 156 DATA 1F0B0F0B0B0601000000 0
      000C1010000000000000 C2C0000 0000000000000000 00000000 000000C0BC6F00000000 00000000
      00000000BF2A :REM*119          00000000000000 :REM*190          00601000000000 :REM*135
111 DATA 0400000000000000BC1400 0 134 DATA 00000000000000000000 0 157 DATA 0000D5B06C1B06060100 5
      00000000000000000000 00000000 00000000000000000000 00000000 5000000000F0540001345 0000000
      00000000000000 :REM*121          00000000000000 :REM*100          00000154000000 :REM*177
112 DATA 00000000000000000000 0 135 DATA 00000000000000000000 0 158 DATA 00000000006A1500000000 0
      00000000000000000000 00000000 00000051000000000000 0055000 0005B150000000000000 0054040
      00000000000000 :REM*48          0000000000045 :REM*150          00000000000000 :REM*76
113 DATA 00000000000000000000 6 136 DATA 00000000000000560000 0 159 DATA 000001050000000010155 5
      01800000000000000000 00000000 00000000FFAA00000000 0000C0F 0000000000000000 :REM*79
      00000000000000 :REM*220          00000000000000 :REM*200          78180602000000 :REM*237
114 DATA 00000000000000000000 0 137 DATA 000900000000000000A0 0 160 DATA 95EA7F958080000056A9 F
      00000000000000000000 00000000 00000000000000000000 00000000 00000000000000000000 0000000
      00000000000000 :REM*54          000A9000000000 :REM*84          6542B020000701CC7B0 2000
115 DATA 00000205000000000000 A 138 DATA 00000055000000000000 0 161 DATA -1 :REM*19
      A5500000000000000AA 5000010 055000000000000000FF 00000000 0000000DF00000 :REM*135
      20A25904000000 :REM*206          00000000000000 :REM*135

```

# Astro-Shoot

Test your marksmanship in this  
space-age shooting gallery.

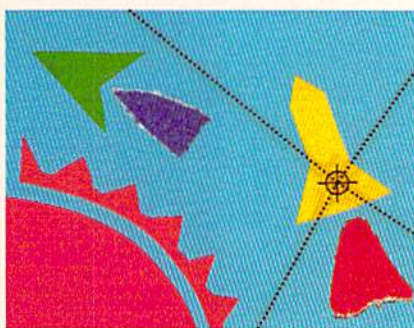


By JOHN FEDOR

**T**wo different types of targets—small and fast-moving, and large and slow-moving—clude you as you shoot from your spaceship in Break-Away. There are up to three large targets on the screen at a time. When you hit one, it breaks into two smaller targets of the same color. If you manage to shoot both of these smaller ones, another large target appears in one corner of the screen.

Since you're in a spaceship, once you start moving, you can't stop. Pressing the fire-button shoots in the direction you're facing, and any shots that don't hit a target are recorded as misses. The ship, controlled by a joystick in port 2, can bounce off the walls with no danger to it or your score.

The game ends when either 60 seconds have elapsed or your ship collides with a target. Then your score is calculated, with five points for each large



target you've hit and ten points for each small one. However, your score is also decreased five points for every miss.

The right side of the screen displays game information, including your present score, your highest previous score, the number of shots you've missed and the time elapsed.

As you play, try to shoot at targets as their paths cross. You'll be more likely

to hit them, and both will be recorded as hits. Also keep in mind that, because misses will decrease your score, constant shooting is unwise. Carefully planned shots, along with skillful maneuvering, will rack up the highest scores.

Break-Away uses six sprite shapes, which I've compiled to save 240 bytes and give you 30 fewer lines of code to type in. Four of the shapes display the ship going in the different directions; the other two are the targets. The bullet is a character graphic (the diamond), to eliminate any collisions between sprites other than your ship and the targets.

This game is fun to play, especially if you're trying to beat a friend. Just don't complain if you make rude comments about his or her flying and marksmanship, but then score a 0 yourself! ☐

*John Fedor breaks away from his college studies with a round or two of this game.*

## Listing 1. Break-Away program.

```

10 POKES3280,0:POKES3281,0:PRIN 252,169,128,133,251,162,0,16 69,3,141,43,208 :REM*183
T" {SHFT CLR}{CTRL 2}BREAK-AW 0,0,189,29,200 :REM*67 160 DATA 141,44,208,169,14,141,
AY IS SETTING UP...PLEASE WA 90 DATA 145,251,200,232,192,24, 45,208,141,46,208,169,1,141
IT." :REM*96 208,245,24,165,251,105,64,13 ,39,208,169,85 :REM*125
20 FORX=49152TO51455:READA:POKE 3,251,165,252 :REM*133 170 DATA 141,21,208,96,169,144,
X,A:NEXT:SYS 49152 :REM*31 100 DATA 105,0,133,252,201,64,2 32,210,255,169,147,32,210,2
30 DATA 169,0,141,74,3,141,75,3 08,226,96,169,0,141,21,208, 55,169,11,141,32 :REM*55
,141,76,3,32,87,195,32,141,1 141,16,208,141 :REM*107 180 DATA 208,141,33,208,169,8,3
92,32,88,192,32 :REM*206 110 DATA 23,208,141,27,208,141, 2,210,255,169,142,32,210,25
40 DATA 10,193,32,246,197,173,3 28,208,141,29,208,169,255,1 5,162,0,169,0 :REM*126
1,208,208,251,173,30,208,41, 41,248,7,162,250 :REM*141 190 DATA 157,0,216,157,192,219,
1,208,244,32,76 :REM*143 120 DATA 142,249,7,232,142,250, 169,67,157,0,4,157,192,7,23
50 DATA 196,32,175,196,32,216,1 7,142,252,7,142,254,7,169,1 2,224,30,208,235 :REM*223
94,32,216,195,32,237,195,32, 43,141,0,208,169 :REM*20 200 DATA 162,0,134,251,134,253,
65,196,32,237 :REM*197 130 DATA 153,141,1,208,160,0,16 169,4,133,252,73,216,133,25
60 DATA 198,32,37,198,32,57,194 2,0,189,173,200,153,4,208,1 4,160,0,152,145 :REM*153
,32,72,199,173,85,3,240,224, 89,176,200,153,5 :REM*244 210 DATA 253,169,66,145,251,160
32,162,196,32,83 :REM*221 140 DATA 208,200,200,200,200,23 29,145,251,169,0,145,253,2
70 DATA 199,32,168,199,32,219,1 2,192,12,208,235,169,5,141, 4,165,251,105,40 :REM*37
99,76,11,192,162,0,138,157,1 65,3,73,15,141 :REM*68 220 DATA 133,251,133,253,165,25
28,62,157,0,63 :REM*201 150 DATA 67,3,169,6,141,69,3,16 2,105,0,133,252,105,212,133
80 DATA 232,208,247,169,62,133, 9,6,141,41,208,141,42,208,1 ,254,232,224,24 :REM*179

```

RUN it right: C-64; joystick

# G A M E S

230 DATA 208,215,169,112,141,0,4,162,110,142,29,4,202,142,192,7,169,125 :REM*32	460 DATA 169,252,141,248,7,174,0,208,172,1,208,173,84,3,32,137,195,141,84 :REM*136	690 DATA 13,56,165,251,233,1,13,3,251,165,252,233,0,133,252,165,2,41,8,240 :REM*191
240 DATA 141,221,7,162,0,160,30,24,32,240,255,162,0,189,179,200,32,210,255 :REM*131	470 DATA 3,140,1,208,142,0,208,96,162,1,142,60,3,142,61,3,142,62,3,142,63 :REM*203	700 DATA 13,24,165,251,105,1,13,3,251,165,252,105,0,133,252,165,251,133,253 :REM*91
250 DATA 232,224,10,208,245,162,1,160,34,24,32,240,255,169,66,32,210,255 :REM*13	480 DATA 3,169,0,141,71,3,141,72,3,141,73,3,157,76,3,232,224,11,208,248 :REM*86	710 DATA 165,252,73,220,133,254,169,0,145,253,177,251,201,32,240,44,120 :REM*92
260 DATA 169,89,32,210,255,162,2,160,30,24,32,240,255,162,0,189,189,200,32 :REM*186	490 DATA 169,96,141,77,3,169,5,141,65,3,141,67,3,141,69,3,96,133,2,41,1 :REM*151	720 DATA 248,24,173,81,3,105,1,141,81,3,173,82,3,105,0,141,82,3,216,88,162 :REM*207
270 DATA 210,255,232,224,10,208,245,162,8,160,30,24,32,240,255,162,0,189 :REM*137	500 DATA 240,1,136,165,2,41,2,240,1,200,165,2,41,4,240,1,202,165,2,41,8 :REM*128	730 DATA 32,142,4,212,232,142,4,212,169,0,141,64,3,141,79,3,141,80,3,76 :REM*122
280 DATA 199,200,32,210,255,232,224,10,208,245,162,12,160,30,24,32,240,255 :REM*39	510 DATA 240,1,232,224,32,176,8,162,31,165,2,73,12,133,2,224,248,144,8,162 :REM*217	740 DATA 113,197,169,90,145,251,165,251,141,79,3,165,252,141,80,3,173,0 :REM*82
290 DATA 162,0,189,209,200,32,210,255,232,224,10,208,245,162,18,160,32,24 :REM*143	520 DATA 248,165,2,73,12,133,2,192,56,176,8,160,55,165,2,73,3,133,2,192 :REM*225	750 DATA 220,41,16,73,16,133,2,240,86,173,83,3,208,81,173,80,3,208,76,56 :REM*242
300 DATA 32,240,255,162,0,189,225,200,32,210,255,232,224,5,208,245,162,22 :REM*58	530 DATA 235,144,8,160,235,165,2,73,3,133,2,165,2,96,206,63,3,173,63,3,240 :REM*106	760 DATA 169,255,237,248,7,170,189,251,200,141,64,3,56,173,0,208,233,24,74 :REM*147
310 DATA 160,32,24,32,240,255,162,0,189,219,200,32,210,255,232,224,6,208 :REM*16	540 DATA 1,96,169,4,141,63,3,169,250,133,251,76,255,195,206,62,3,173,62,3 :REM*115	770 DATA 74,74,141,79,3,56,173,1,208,233,50,74,74,74,168,169,4,141,80,3 :REM*19
320 DATA 245,162,0,169,1,157,134,217,157,22,219,157,38,218,157,182,219,232 :REM*46	550 DATA 240,1,96,169,12,141,62,3,169,251,133,251,162,0,134,252,134,253 :REM*240	780 DATA 192,0,240,20,24,173,79,3,105,40,141,79,3,173,80,3,105,0,141,80,3 :REM*218
330 DATA 224,10,208,239,32,57,194,32,89,194,32,121,194,32,171,194,96,162,0 :REM*144	560 DATA 166,252,189,250,7,197,251,208,38,189,65,3,133,2,164,253,185,4,208 :REM*54	790 DATA 136,208,232,162,128,142,4,212,232,142,4,212,32,216,197,165,2,141 :REM*200
340 DATA 160,3,185,70,3,74,74,74,4,74,9,48,157,136,5,232,185,70,3,41,15,9,48 :REM*255	570 DATA 170,185,5,208,168,165,2,32,137,195,134,2,166,252,157,65,3,152,164 :REM*1	800 DATA 83,3,96,173,79,3,133,251,173,80,3,133,252,160,0,177,251,201,32 :REM*149
350 DATA 157,136,5,232,136,208,229,96,162,0,160,3,185,73,3,74,74,74,74,9 :REM*171	580 DATA 253,153,5,208,165,2,153,4,208,230,252,166,253,232,232,134,253,224 :REM*22	810 DATA 240,11,169,0,141,64,3,141,79,3,141,80,3,96,162,0,138,157,0,212 :REM*65
360 DATA 48,157,40,6,232,185,73,3,41,15,9,48,157,40,6,232,136,208,229,96 :REM*79	590 DATA 12,208,197,96,162,4,160,0,136,208,253,202,208,250,96,173,20,3,133 :REM*161	820 DATA 232,224,24,208,248,169,15,141,24,212,169,38,141,5,212,169,32,141 :REM*99
370 DATA 173,77,3,74,74,74,74,9,48,141,24,7,173,77,3,41,15,9,48,141,25,7 :REM*165	600 DATA 113,173,21,3,133,114,169,0,141,86,3,120,169,196,141,21,3,169,104 :REM*177	830 DATA 1,212,169,116,141,12,212,169,18,141,8,212,169,83,141,19,212,169,6 :REM*203
380 DATA 169,46,141,26,7,173,78,3,74,74,74,74,9,48,141,27,7,173,78,3,41,15 :REM*164	610 DATA 141,20,3,88,96,238,86,3,173,86,3,201,3,208,39,169,0,141,86,3,120 :REM*8	840 DATA 141,15,212,96,173,31,208,41,252,133,2,208,1,96,173,79,3,133,251 :REM*22
390 DATA 9,48,141,28,7,96,173,82,3,74,74,74,74,9,48,141,18,5,7,173,82,3,41 :REM*37	620 DATA 248,56,173,78,3,233,5,141,78,3,173,77,3,233,0,141,77,3,208,13,173 :REM*146	850 DATA 173,80,3,133,252,160,0,169,32,145,251,173,31,208,140,80,3,140,79 :REM*117
400 DATA 15,9,48,141,186,7,173,81,3,74,74,74,74,9,48,141,187,7,173,81,3,41 :REM*235	630 DATA 78,3,208,8,169,1,141,85,3,32,162,196,32,121,194,32,171,194,108 :REM*223	860 DATA 3,169,4,133,251,132,252,165,2,37,251,240,7,173,21,208,37,2,208,3 :REM*78
410 DATA 15,9,48,141,188,7,96,206,60,3,173,60,3,240,1,96,169,8,141,60,3 :REM*128	640 DATA 113,0,120,165,113,141,20,3,165,114,141,21,3,88,96,206,61,3,173,61 :REM*201	870 DATA 76,223,198,162,32,142,11,212,232,142,11,212,166,252,189,250,7,201 :REM*62
420 DATA 173,0,220,41,15,73,15,133,2,41,1,240,15,173,84,3,41,12,9,1,141,84 :REM*49	650 DATA 3,240,3,76,113,197,169,6,141,61,3,173,79,3,133,251,173,80,3,133 :REM*245	880 DATA 250,240,43,169,250,157,250,7,157,251,7,189,65,3,73,15,157,66,3 :REM*23
430 DATA 3,169,255,141,248,7,165,2,41,2,240,15,173,84,3,41,12,9,2,141,84,3 :REM*105	660 DATA 252,208,3,76,113,197,160,0,169,32,145,251,173,64,3,133,2,41,1,240 :REM*166	890 DATA 138,10,168,185,4,208,153,6,208,185,5,208,153,7,208,8,165,251,10,13 :REM*206
440 DATA 169,253,141,248,7,165,2,41,4,240,15,173,84,3,41,3,9,4,141,84,3 :REM*177	670 DATA 13,56,165,251,233,40,133,251,165,252,233,0,133,252,165,2,41,2,240 :REM*247	900 DATA 21,208,141,21,208,76,194,198,165,251,73,255,45,21,208,141,21,208 :REM*78
450 DATA 169,254,141,248,7,165,2,41,8,240,15,173,84,3,41,3,9,8,141,84,3 :REM*22	680 DATA 13,24,165,251,105,40,133,251,165,252,105,0,133,252,165,2,41,4,240 :REM*246	910 DATA 120,248,24,173,71,3,105,5,5,141,71,3,173,72,3,105,0,141,72,3,173,73 :REM*63

# GAMES

920 DATA 3,105,0,141,73,3,216,8 8,120,248,24,173,71,3,105,5 141,71,3,173,72 :REM*227	1020 DATA 82,3,56,173,71,3,233, 5,141,71,3,173,72,3,233,0, 141,72,3,173,73 :REM*189	1120 DATA 128,0,63,0,0,192,0,0, 224,0,0,240,0,0,255,128,0, 255,128,0,240,0 :REM*111
930 DATA 3,105,0,141,72,3,173,7 3,3,105,0,141,73,3,216,88,6 251,230,252,165 :REM*100	1030 DATA 3,233,0,141,73,3,216, 176,184,169,0,141,71,3,141 72,3,141,73,3 :REM*83	1130 DATA 0,224,0,0,192,0,0,255 192,0,127,128,0,63,0,0,30 0,0,12,0,0,12,0 :REM*234
940 DATA 252,201,6,208,1,96,76, 78,198,173,21,208,41,12,208 23,169,251,141 :REM*156	1040 DATA 240,171,173,76,3,205, 73,3,144,24,240,2,176,38,1 73,75,3,205,72,3 :REM*9	1140 DATA 0,12,0,0,0,0,1,128, 0,3,128,0,7,128,0,255,128, 0,255,128,0,7 :REM*170
950 DATA 250,7,169,32,141,4,208 169,56,141,5,208,173,21,20 8,9,4,141,21,208 :REM*106	1050 DATA 144,12,240,2,176,26,1 73,74,3,205,71,3,176,18,17 3,71,3,141,74,3 :REM*167	1150 DATA 128,0,3,128,0,1,128,0 12,0,0,12,0,0,12,0,0,30,0 0,63,0,0,127 :REM*49
960 DATA 173,21,208,41,48,208,2 3,169,251,141,252,7,169,32, 141,8,208,169 :REM*204	1060 DATA 173,72,3,141,75,3,173 73,3,141,76,3,96,32,89,19 4,173,0,220,41 :REM*153	1160 DATA 128,0,255,192,0,0,0, 32,247,32,56,56,235,66,82 69,65,75,45,65 :REM*95
970 DATA 235,141,9,208,173,21,2 08,9,16,141,21,208,173,21,2 08,41,192,208,23 :REM*97	1070 DATA 16,240,249,162,12,160 5,24,32,240,255,162,0,189 230,200,32,210 :REM*205	1170 DATA 87,65,89,74,79,72,78, 32,70,69,68,79,82,76,65,83 84,32,83,67,79 :REM*131
980 DATA 169,251,141,254,7,169, 32,141,12,208,169,247,141,1 3,208,173,21,208 :REM*147	1080 DATA 255,232,224,21,208,24 5,173,0,220,41,16,208,249, 32,1,193,173,0 :REM*249	1180 DATA 82,69,72,73,71,72,32, 83,67,79,82,69,77,73,83,83 69,68,84,73,77 :REM*64
990 DATA 9,64,141,21,208,96,173 30,208,41,1,240,3,141,85,3 96,169,255,141 :REM*194	1090 DATA 220,41,16,240,249,96, 32,65,196,32,65,196,162,0, 160,0,136,208 :REM*23	1190 DATA 69,82,5,80,82,69,83,8 3,32,66,85,84,84,79,78,32, 84,79,32,80,76 :REM*67
1000 DATA 27,208,32,171,194,32, 57,194,32,12,200,173,81,3, 208,6,173,82,3 :REM*190	1100 DATA 253,202,208,250,96,0, 0,0,0,0,0,0,0,0,12,0,0,30, 0,0,12,0,0,0,0,0 :REM*187	1200 DATA 65,89,1,4,2,8,239 :REM*107
1010 DATA 208,1,96,120,248,56,1 73,81,3,233,1,141,81,3,173	1110 DATA 0,0,0,0,0,0,63,0,0,97	

## Faster than a Speeding Cartridge More Powerful than a Turbo ROM

It's Fast, It's Compatible, It's Complete, It's...

# JiffyDOS™

Ultra-Fast Disk Operating System for the C-64, SX-64 & C-128

- **Speeds up all disk operations.** Load, Save, Format, Scratch, Validate, access PRG, SEQ, REL, &USR files up to 15 times faster!
- **Uses no ports, memory, or extra cabling.** The JiffyDOS ROMs upgrade your computer and drive(s) internally for maximum speed and compatibility.
- **Guaranteed 100% compatible with all software and hardware.** JiffyDOS speeds up the loading and internal file-access operation of virtually all commercial software.
- **Built-in DOS Wedge plus 14 additional commands and convenience features** including one-key load/save/scratch, directory menu and screen dump.
- **Easy do-it-yourself installation.** No electronics experience or special tools required. Illustrated step-by-step instructions included.

JiffyDOS is available for C-64, 64C, SX-64, C-128 & C-128D (JiffyDOS/128 speeds up both 64 and 128 modes) and 1541, 1541C, 1541-II, 1571, 1581, FSD-1&2, MSD-1&2, Excel 2001, Enhancer 2000 disk drives. System includes ROMs for computer and disk drive, stock/JiffyDOS switching system, illustrated step-by-step installation instructions, User's Manual, Money-Back Guarantee, & unlimited customer support.

C-64/SX-64 systems \$49.95; C-128/C-128D systems \$59.95. Add 1 drive ROM's \$24.95. Please add \$4.25 shipping/handling per order. VISA/MC. COD. Money Order accepted. Call or write for more information. Dealer, Distributor, & Users' Group pricing available. Please specify computer and drive when ordering.

### Creative Micro Designs, Inc.

P.O. Box 789, Wilbraham, MA 01095 Phone: (413) 525-0023  
50 Industrial Dr., Box 646, E. Longmeadow, MA 01028 FAX: (413) 525-0147

Circle 418 on Reader Service card.

## WE WON'T PAY YOUR TAXES!

But **TAX MASTER** will help you compute them more QUICKLY and EASILY. Be the Master of your Income Taxes with **TAX MASTER**, now available for your 1988 Federal Income Taxes for the C-64/C-128 with single, twin or dual disk drive and optional printer.

- NEW Tax laws are covered.
- **FORMS 1040, 4562, & Schedules A, B, C, D, E & F.**
- **PERFORMS** all arithmetic CORRECTLY.
- **EASY CHANGE** of any entry with automatic **RECALCULATION** of the entire form.
- **TRANSFERS** numbers between forms.
- **CALCULATES** your taxes and **REFUND**. Tax tables are included.
- **SAVES** all your data to disk for future changes.
- **PRINTS** the data from each form.
- **CALCULATOR** function is built-in.
- **DISCOUNT** coupon toward the purchase of next year's updated program is included.

**TAX MASTER . . . . . (ON DISK) . . . . . ONLY \$32.00**

## TIRED OF SWITCHING CABLES?

**VIDEO MASTER 128** provides continuous 80 column color (RGBI), 80 column monochrome and audio out. Switch between 80 column monochrome and 40 column color for composite monitor. Use up to 4 monitors at once! Includes composite cable.

**VIDEO MASTER 128 . . . . . for Commodore 128 . . . . . \$39.95**

## FED UP WITH SYNTAX ERRORS?

**HELP MASTER 64** provides instant On-Line Help screens for all 69 BASIC commands when you need them. Takes no BASIC RAM. No interference with loading, saving, editing or running BASIC programs. Includes 368 page BASIC reference text, more.

**HELP MASTER 64 . . . . . for Commodore 64, 64C . . . . . \$24.95**

## OTHER MASTER SOFTWARE ITEMS

<b>RESET MASTER C-64</b> (not 64C) reset switch w/2 serial pts	\$ 24.95
<b>CHIP SAVER KIT</b> protects computer's chips from static	5.95
<b>MODEM MASTER</b> user port extender \$29.95; w/reset	34.95
<b>Y-NOT?</b> 6-foot serial Y cable, 1 male, 2 female connectors	15.00
<b>Y-YES!</b> 6-foot serial Y cable, 3 male connectors	15.00
<b>C-128 80 col. monochrome cable</b> for non-RGB monitor	9.00
<b>Disk Notcher</b> —lets you use both sides of disk	6.00
<b>64-TRAN</b> The only Fortran compiler for C-64/64C	50.00

## Send for Free Catalog

# MASTER SOFTWARE

6 Hillery Ct.  
Randallstown, MD 21133  
(301) 922-2962

ADD \$2.00 per order shipping & handling US and Canada, \$7.00 foreign. All prices in US Dollars. Canadian orders use Canadian POSTAL money order. Maryland residents add 5% tax. Dealer inquiries welcome!



Circle 414 on Reader Service card.

# Commodore Clinic

Here's a panorama of 1988's most useful questions and answers  
in the areas of programming, software and hardware.



By LOU WALLACE

## PROGRAMMING

**Q** I'm a 13-year-old programmer and I think your magazine is great! I'm writing a program and I want to know how to keep the listing from prying eyes.

—BRIAN LEABY  
N. BABYLON, NY

**A** On the C-64, use POKE 775,1 to disable program listing capability. To restore it, use POKE 775,167. On the C-128, use POKE 775,139 to disable and POKE 775,81 to restore.

**Q** I accidentally used Save with Replace (SAVE "@0:FILENAME",8) on a C-64 program when I typed it in for the first time, not knowing I shouldn't have. Now I can't get the filename off the disk! Can you tell me how? Also, should I make a backup before scratching the program?

—NANCY A. KUSCHE  
GLENDALE, AZ

**A** The easiest way is to simply rename the file. Use this command line in Direct mode:

```
OPEN 15,8,15,"R0:NEWFILENAME=  
0:@0:FILENAME":CLOSE 15
```

Now you can scratch the old file without worrying about erasing the program with the filename "Filename".

**Q** I've been looking for the C-128 Programmer's Reference Guide in every bookstore in town, but no one has heard of it. Does it really exist, and can you tell me where I can order one?

—KEN JAMERSON  
ZANESVILLE, OH

**A** Oh yes, it certainly exists and is perhaps the single most important programming reference available for

the C-128 (I usually carry one in my briefcase—honest!). It's over 700 pages long, contains a complete Basic 7 encyclopedia, sections on machine language, graphics, sprites, sound and CP/M. It also covers programming the 8563 80-column chip, a device input/output guide, memory maps and nearly 100 pages of hardware specifications.

It's published by Bantam Computer Books and retails for \$24.95. You can order a copy by calling 800-223-6834, extension 479. Its reference number is 34378-5, and they accept credit cards (VISA, MasterCard and American Express). You can also get information for ordering by mail from that number.

**Q** What is the difference between the space and shifted space characters?

—JOSE RAMIREZ  
GUADALAJARA, JALISCO, MEXICO

**A** The space character is a CHR\$(32), while shifted space is a CHR\$(160). They are different ASCII characters and should not be used interchangeably. For example, if you were typing in a RUN program listing and typed in a shifted space instead of a space, RUN's Checksum program would generate an error, but the line would look perfectly correct to you, making it very difficult for you to debug.

**Q** I'm writing a program and want to keep a few files open while the program is running. How many can I keep open at one time on a Commodore disk drive?

—D. SCHWARTZ  
SCOTTSDALE, AZ

**A** It depends on the drive you use. 1541s and 1571s have three available buffers to open up to three sequential files at once, or one relative file (which requires two buffers) and

one sequential file (one buffer). The 1581 drive has seven buffers, so you can have more combinations of open files, but it's not a good idea to have more than two relative files open at once.

**Q** I need information about both the inner workings of the C-128's ROM routines and those of the 1571 disk drives. Does Commodore sell documented ROM listings? If so, where can I find them and how much does it cost?

—MARY SUE JENNINGS  
TULSA, OK

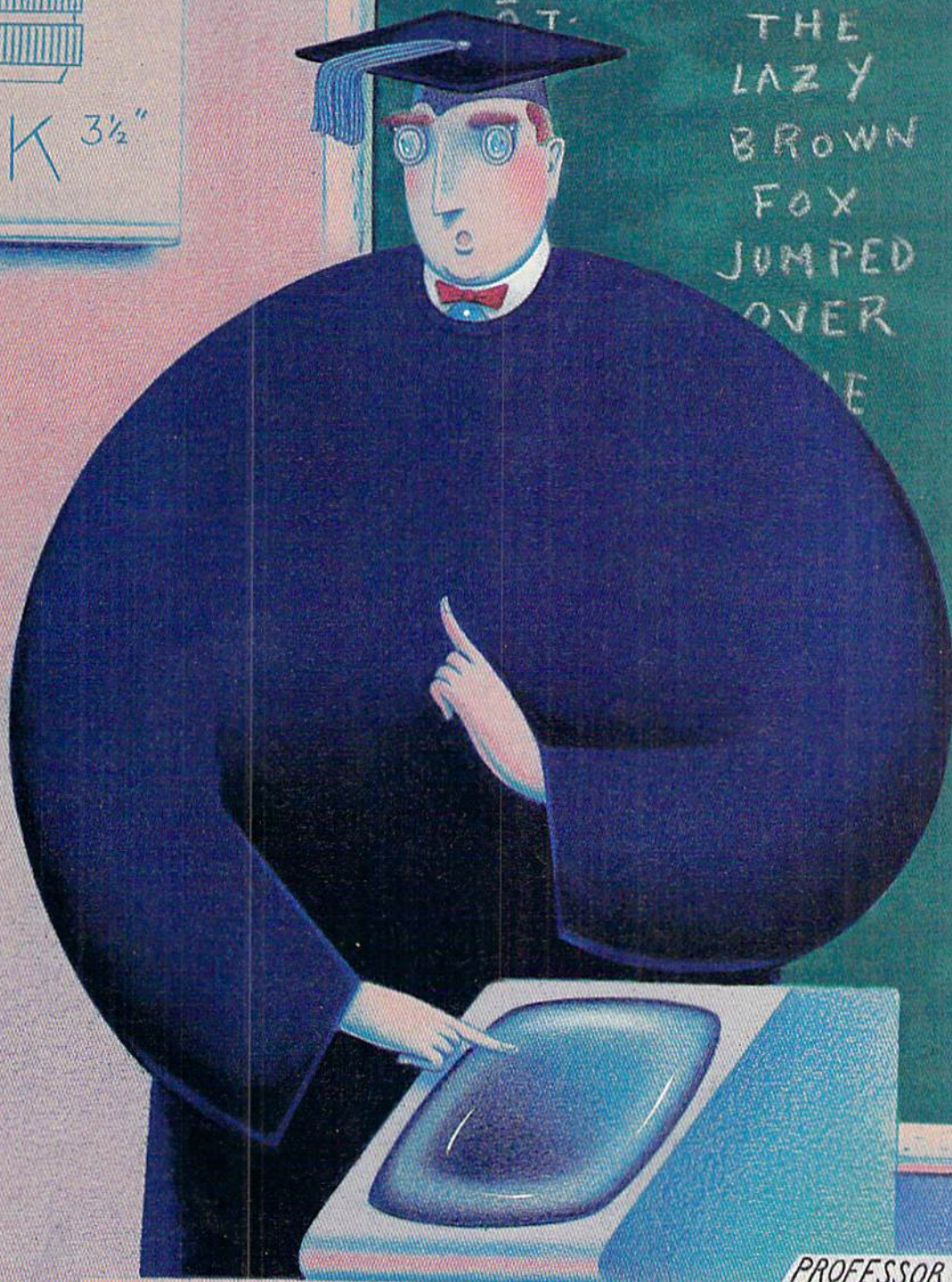
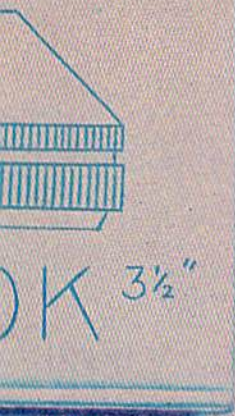
**A** Commodore doesn't sell that information as far as I know, but you can get books with exactly that information from Abacus Software of Grand Rapids, Michigan (phone 616-241-5510). They have the 1571 Disk Drive Internals (\$19.95) and the C-128 Basic 7.0 Internals (\$24.95), which has extensive documentation on ROM routines. Both are excellent reference books on the inner workings of 128 Basic and the 1571 ROMs. Abacus also publishes a number of books for the 64 and 128.

**Q** I'm taking a course in Basic, and the class uses MS-DOS computers. However, I want to do my homework on my C-64. The problem is Basic 2.0 doesn't have any text-formatting abilities, and I need to have columns of decimal numbers to two significant digits, with the decimal points aligned. How can I do it?

—M. WRIGHT  
ODUM, GA

**A** It's not too difficult. All you need is a short routine to read in a number, truncate it to two digits to the right of the decimal, and convert it to a string. Once that's done, you can add extra zeros if the number has fewer than two significant digits, and pad it with ▶





READIN'

WRITIN'

COMPUTIN'

AT:

THE  
LAZY  
BROWN  
FOX  
JUMPED  
OVER  
THE

10 REM  
20 POKE  
30 S=542  
40 POKEST  
50 IFX=D  
60 NEXT:  
70 IFCS<7  
80 PRINT"  
90 DATA84  
100 -----  
110 -----

PROFESSOR COMMODORE

enough space characters so all the numbers have the same number of digits. This short Basic program segment demonstrates one way to do it:

```
10 PRINT "X=";:INPUT Y
20 GOSUB 50
30 PRINT Y,X$
40 GOTO 10
50 X=Y*100:X=INT(X):X=X/100
60 X$=STR$(X):T=X-INT(X):IF T=0
  THEN X$=X$+".00":GOTO 80
70 T=(X*10)-INT(X*10):IF T=0 THEN
  X$=X$+"0":GOTO 80
80 FOR J=1 TO (10-LEN(X$)):X$=" "+X$:
  NEXT
90 RETURN
```

In this example, the program asks you for a number and converts the number to a string with the proper number of digits. It then adds character spaces to the front of the number so the output is ten characters long, with the decimal points aligned. You can change the number of zeros the program adds to a number by changing the 10 in line 80. And the subroutine at lines 50-90 could be included in your programs.

However, while writing this, I discovered a bug in both the C-64 and C-128 math routines and thought I would pass it on to our readers. Apparently it involves round-off errors in floating-point math, but only affects certain numbers, for which I haven't as yet been able to find a pattern. It's easier to demonstrate than explain. In Direct mode, type in the following lines:

```
X=33.4
Y=(X*10)-INT(X*10)
PRINT Y
```

What do you think the value of Y is? If you answer zero, you're right, but the answer you get is 1.1920929E-07! Not quite right, that's for sure. If you had entered -33.4 for X, the value given for Y would have been a little less than 1, a very large error indeed. These errors are enough to cause programs to work incorrectly when they encounter one of the "magic" error-causing numbers, as in the small routine above. So be warned!

**Q** I was intrigued by your numeric format program in the March 1988 Commodore Clinic. I've made some enhancements to it and corrected a couple of minor bugs that were due to the math bug you described. The result is an improved version you might be interested in passing on to your readers.

```
10 PRINT "X=";:INPUT Y
20 GOSUB 50
30 PRINT Y,X$
```

```
40 GOTO 10
50 X=Y*100:X=INT(X+.0001):X=X/100
60 X$=STR$(X):T=X-INT(X):IF T=0
  THEN X$=X$+".00":GOTO 80
70 V$=STR$(INT(X*10+0)):U=
  VAL(V$):S$=STR$(X*10):R=VAL(S$)
75 Z=R-U:IF Z=0 THEN X$+X$+"0"
80 FOR J=1 TO (10-LEN(X$)):X$=" "+
  X$:NEXT
90 RETURN
```

—HERBERT WALLER  
HICKSVILLE, NY

**A** Thanks, Herb. The original formatting program generated quite a few letters, and a number of readers sent along their own versions. Evidently a good many have also been experimenting with the math bug. If anyone solves the math-bug problem, drop us a line.

**Q** I'm writing to you in reference to a "bug" in the Commodore math routines you discussed back in the March 1988 Clinic. The problem isn't a bug, but a consequence of the floating-point math routines. It results from rounding off, as you pointed out, and from the rounding of answers in the floating-point routines.

In your example, the rounding that contributes to the problem occurs in X\*10 and is a result of the representation of decimal 0.4 in the binary floating-point system, which causes the shift of a 1 into bit 7 of the rounding byte. The results for the negative case are essentially the same except for the operation of the INT(...) function of Basic, which returns the closest integer value that is less than the argument. For negative numbers, this gives a (negative) integer that is larger in absolute value (magnitude without regard to sign) than the argument.

—STUART RUDIN, PHD  
SCOTTSDALE, AZ

**A** Thanks for the information. As I've stated before, this is something all Basic programmers who use floating-point routines should be aware of. Interestingly, this "math-bug" subject has generated more discussion than the alleged use of astrology in the White House. Still, the more we know about our computers, the better programmers and users we become.

**Q** I just bought the January 1988 issue of RUN, and I ran into difficulty trying to use the new checksum. I get an Out of Data error in line 30. I checked each line to see if I'd made any typing errors but could find none.

Please advise me, as I would like to type

in the Calendar Maker program, among others in your magazine.

—JANIS SUNKEN  
NEVADA, IA

**A** You probably made an error in typing, as RUN's new Checksum program works very well. But, just to be sure, I typed it in myself from the listing in the January '88 issue. No problem! It worked as it should. Since I don't have a copy of your listing, I can't be sure what is wrong. But the computer has given us a good hint with the "Out Of Data error in line 30" message. That tells us the program was trying to read a value from the Data statements and ran out of data to read, which in turn means you left out a portion of a Data statement. That could be a missing comma, a period used in place of a comma, or perhaps even a complete line. Check the listing again carefully; I'm sure you will discover the problem.

**Q** I know that Apple computers have a Basic command called Speed, which sets the speed of output to the screen or printer. And I have noticed that in Mediagenic's Hacker and Ghostbusters, the Speed command is available, but I cannot find any way of accessing it in CBM Basic. Does it exist on the C-64?

—CHAD HAYNES  
BECKLEY, WV

**A** No, there is no corresponding command for the C-64 (or the C-128). What the Mediagenic programs do is slow down or speed up the screen output itself. You can emulate it quite easily in your programs, too. Just put a small delay loop between outputs, thusly:

```
100 PRINT A$:FOR T=1 TO DE:NEXT
110 GOTO 100
```

By increasing the value of the variable DE in line 100, you can slow down the speed of printing A\$. If you decrease DE, it will print faster.

**Q** How can I reprogram the C-128's shifted run key and the help key? Also, how can I keep a user from exiting a Basic 7.0 program with the stop key?

—R. BURNS  
CONCORD, NH

**A** The answers to both questions were published recently in the Best of Magic column in RUN's Special Issue #4. To disable run-stop/restore, use these Pokes:

```
POKE 792,51:POKE 793,255
```

## COMMODORE CLINIC

As most 128 users know, the eight function keys can easily be reprogrammed with the Key command. But you can easily reprogram the shifted run and help keys with the following SYS commands:

```
RUN KEY
BANK 15:SYS DEC("60EC"),8,,"RUN KEY"
HELP KEY
BANK 15:SYS DEC("60EC"),9,,"HELP KEY"
```

Substitute your own commands for the words RUN KEY or HELP KEY to give you a total of ten programmable function keys.

**Q** *When I'm creating high-resolution graphics on my C-64, I have a problem positioning pixels of different colors next to each other—the first colored dot changes to the new color! Why does that happen, and what can I do about it?*

—LEE RUMSEY  
LONG BEACH, CA

**A** The problem you're experiencing is known as "color bleed." The way C-64 (and C-128) graphics are generated by the VIC chip allows two colors per color cell, one foreground and one background color. The color cell is an 8x8-pixel matrix. So, any attempt to change even one dot within the 8x8 area results in all foreground pixels (or all background pixels, depending on which color you're using) changing to the current color. There's nothing you can do about it when using the hi-res 320x200 Graphics mode.

If you use Multicolor Graphics mode, you can have up to four different colors per cell, but there are also some drawbacks to using this mode. The screen resolution reduces to 160x200, giving a coarser display, and the color cell becomes horizontally smaller—a 4x8-pixel matrix. Each pixel has twice the width as in Hi-Res mode.

**Q** *What do you do when you've played one C-64 game, and you want to reset and play a different game without using the on/off switch? The reason I ask is that I want to use the switch as little as possible. Also, what is a Poke statement in a program for, and how do you use it in a C-64 program?*

—NGHIA LUONG  
SANTA ANA, CA

**A** Sometimes you can press the run-stop/restore key combination, and then enter the command SYS 64738 to perform a "soft" reset. However, most games disable the run-stop/restore com-

bination, so that a soft reset won't work. An alternative is to add a hardware reset switch similar to that of the C-128. The easiest way to do that is to use a plug-in cartridge (like Power Cartridge or Blowup). But even then, some games install themselves in memory like an autostart cartridge, so any attempt to reset the computer merely restarts the game. In that event, you can only turn the computer off, wait about five seconds and turn it back on.

The Poke command is a way for the Basic programmer to place a number into a specific memory location in the computer's memory. The number must be in the range 0-255, as 255 is the largest value that can be contained in one byte of memory. To use it, just type POKE, followed by the memory address, a comma and then the value to place in that address. For example, if you wanted to put the character A onto the C-64 screen, you could type:

```
POKE 1024,65
```

This places the ASCII value 65 (the code for letter A) into memory location 1024, which is the first character cell, for the upper-left corner of the C-64 screen.

Related to the Poke command is the Peek command. This does the opposite, reading the contents of a memory address. For example,

```
A = PEEK(1024)
```

assigns to variable A the ASCII value of the byte at address 1024.

**Q** *I've tried saving multicolor graphic screens (GRAPHIC 3) with my C-128 and have had no luck. All I can do is save the bitmap and one color; the other colors never show up right. According to Commodore, these colors are in the color memory area at 55296-56295. But saving that area of memory and reloading it has no effect on the colors of the images. Is there a way to save multicolor pictures from Basic, and if so, could you please let us in on the secret?*

—MORGAN HAUEISEN  
HAMILTON, NJ

**A** Yes, it can be done from Basic, but if I tell you, you'll have to promise to keep our "secret" safe! There are three areas of memory you must save in order to completely reproduce a multicolor (GRAPHIC mode 3) screen on the C-128. The first is the area from 7168 to 16191, which contains the same color (7168-8191) and bitmap (8192-16191). You must also save the background color at 53281. (I also save the

border at 53280.) And, as you know, the main color memory is at location 55296-56295. The trick is to be able to access that memory correctly, because in the C-128's multi-bank memory scheme, some areas are not accessible (even using the Bank command) without a little nudge. To demonstrate, I wrote a small program that creates a multicolor display, then saves it to disk. A second routine can be used to display the picture. You could easily use these as subroutines in your own programs for loading and saving pictures.

```
5 REM SAVE MULTICOLOR SCREEN
10 POKE 53280,7
20 COLOR 0,1:COLOR 1,2:COLOR
   2,3:COLOR 3,4
30 REM SAVE MC PICTURES
40 GRAPHIC3,1
50 REM CREATE SOMETHING
60 CIRCLE 1,40,100,35
70 CIRCLE 2,80,100,35
80 CIRCLE 3,120,100,35
90 REM SAVE IT
100 POKE 1,PEEK(1)AND254
110 BSAVE"BKGBDR",B13,P53280 TO
   P53282
120 BSAVE"CMEM",B15,P55296 TO
   P56296
130 BSAVE"SMEM - BMP",B0,P7168 TO
   P16192
140 GETKEY A$
150 GRAPHIC0
```

Once you've saved your pictures to disk, you can reload and display them with this routine:

```
10 REM RELOAD A SAVED MULTICOLOR
   PICTURE
20 GRAPHIC3,1
30 POKE 1,PEEK(1)AND254
40 BLOAD"BKGBDR",B13,P53280
50 BLOAD"CMEM",B15,P55296
60 BLOAD"SMEM - BMP",B0,P7168
70 GETKEY A$
80 GRAPHIC0
```

**Q** *I've written a C-128 program that needs to know the name of the 1571 disk so it will be able to print out the disk name, along with other data, to the printer. The trouble is, I can't figure out how to do it! Any ideas?*

—JOHN SCHUELER  
SEDONA, AZ

**A** It's quite easy, and there are a number of ways to do it. The easiest is to open a channel to the drive and read the directory, just as you would a program. Here's a short Basic program that reads the disk name into a variable. With a little more work, it ▶

# COMMODORE CLINIC

could probably be used to read the entire disk directory, along with all the file types and file sizes.

```
10 REM READ A DISK NAME
20 OPEN 1,8,0,"$"
30 FOR I=0 TO 24
40 GET#1,A$:IF A$="" THEN A$=
  CHR$(0)
50 D$=D$+A$:NEXT
60 CLOSE 1
70 D$+MID$(D$,9,16)
80 PRINT D$
```

This works on any 1541, 1571 or 1581 drive. It reads in enough information to get just the disk name, then, using the MID\$ function, it makes D\$ equal to the data that's found between the double quotes at the top of your disk directory. This is the disk name, and it's always 16 characters long.

**Q** What is the purpose of the REM\* statements at the end of each line of code in programs listed in RUN?

—Y. RUBINSON  
DES PLAINES, IL

**A** They're used with RUN's Checksum program, which catches errors you may type into a listing. Those REM\* statements are followed by a number, called a checksum. Don't type in the REM\* or the number that follows. Instead, when you have the Checksum program installed in memory and type in a program line, a number is printed to the screen. If the number matches the number following the REM\*, you've correctly entered that line; if not, there's an error in the line that needs correcting. The Checksum program replaced our old Perfect Typist program in January 1988. It and the instructions for using it appear in every issue.

**Q** I'm a beginning machine language programmer, and I'd like to learn how to write an interrupt-driven program for the C-64. Could you give me a simple example?

—JANE ROCKMYER  
CLEVELAND, OH

**A** I sure can. Writing an IRQ routine is basically very simple. In theory, all you have to do is redirect the IRQ vectors at \$0314 to the address of your routine. Then, every 60th of a second, your machine language program is executed. After each access, it should then send the computer on to the address of the regular IRQ routine. To demonstrate, I wrote a very simple program that changes the border and back-

ground colors to black and cyan, respectively. You can poke in a new value for the border color (53280) or the background color (53281), but all you'll get is a momentary flash, and the colors instantly return to black and cyan. Here's the machine language source code for the routine.

```
* = $C000 ; the code is
                placed at 49152
                decimal
```

```
border = 53280
background = 53281
irqvec = $0314
irqold = $EA31 ; this is the normal address
                found in
                $0314/$0315
```

```
init          sei
              lda #<irq
              ldy #>irq
              sta irqvec
              sty irqvec + 1
              cli
              rts

irq           sta tempa ; store a, x
              ; and y

              stx tempx
              sty tempy
              lda bdcolor ; border color
              sta border
              ldx bkcolor ; background
              ; color

              stx background
              lda tempa ; restore a, x
              ; and y

              ldx tempx
              ldy tempy
              jmp irqold

tempa        .byt 0
tempx        .byt 0
tempy        .byt 0
bdcolor      .byt 0 ; black border
bkcolor      .byt 3 ; cyan back-
              ; ground

              .end
```

If you'd like to see what it does, just type in the short Basic loader below. Once run, it places the machine language routine at 49152 and activates it with a SYS call. The border becomes black and the background cyan. Try changing the colors with Pokes to 53280 (border) and 53281 (background). To get rid of the effect, press the run-stop/restore keys.

```
10 REM SIMPLE IRQ DEMO
20 REM LOU WALLACE
30 REM RUN MAGAZINE
40 AD = 49152
50 READ A:IF A = -1 THEN SYS
  49152:END
```

```
60 POKE AD,A:AD = AD + 1
70 GOTO 50
80 DATA 120,169,13,160,192,141,20,3
90 DATA 140,21,3,88,96,141,46
100 DATA 192,142,47,192,140,48,192
110 DATA 173,49,192,141,32,208,174
120 DATA 50,192,142,33,208,173,46
130 DATA 192,174,47,192,172,48,192
140 DATA 76,49,234,0,0,0,0
150 DATA 3, - 1
```

**Q** I have a C-128, 1541 and Okidata 120 printer. I'm pretty new to computing, and I can't figure out how to list a Basic 7.0 program to my printer. I've tried the Open command, but it isn't working quite right. What I need is explicit instructions!

—G. STOJHOVIC III  
PHOENIX, AZ

**A** As Alf would say, "No problem!" All you need to do is load the Basic program, then, with your printer on, type the following in Direct mode. (Direct mode means just type it and press return.)

```
OPEN 4,4,7:CMD 4:LIST:PRINT#4:CLOSE 4
```

Your program will be printed on your printer, just as it appears on the screen. The only exceptions will be Quote-mode commands, which, depending on your printer's interface, will be translated into either ASCII sequences or graphics characters.

**Q** Can Commodore 64 and 128 programs be converted to run on an IBM clone XT, with MS-DOS and GW Basic?

—T. WILDER  
PETERBOROUGH, NH

**A** If the programs are simple Basic programs, it is possible to convert them. However, if they use graphics, sprites, sound commands or any other machine-specific abilities, the conversion problems become significant, since IBM clones have limited sound, no sprites and different graphics resolutions. Also, you will have to rewrite any disk accesses, as they use different commands and techniques.

**Q** Has the "garbage collecting" problem when using large arrays on the C-64 been corrected in the C-128?

—R. S. DEFREITAS  
LAKE HAVASU CITY, AZ

**A** Yes, it has. Since the 128 has two 64K RAM banks, one is dedicated to the Basic text (bank 0) and the other

to Basic variables (bank 1). On the 128, each string has a "pointer" to the variable using it, which makes garbage collection (which means to discard unused strings and compact the memory required to store them, freeing it for other uses) much, much faster than on the 64. Essentially, the 64 has to search the entire variable list for matches to the strings in order to perform garbage collecting. The 128's method is possible only because of the large amount of RAM available for variables. The 64 doesn't have that luxury, so it uses the slower but more efficient method.

**Q** *I've been writing an adventure game in Basic on my C-128. I've gotten to the stage where I'm able to play it, but after a certain number of entries, I get an Out of Memory message.*

*My manual states, "Either there is no more room for program code and/or program variables, or there are too many nested Do, For or Gosub statements in effect."*

*I assume that there are too many For-Next loops, because my program uses these to determine its next action. I've tried using the Trap statement, but that only results in a computer lockup.*

*Have you any suggestions?*

—R. JAKLITSCH  
WICKLIFFE, OH

**A** There are a couple of possible problems. First, you may be out of variable memory. Even though there's a 64K bank in the 128 set aside for variables, it's not difficult to run out of memory, if you use enough variables. Consider this DIMENSION statement:

```
DIM A$(21414)
```

This allocates enough memory for 21,414 string variables in the array A\$( ), leaving four bytes free. Increase it to 21,415, and you get an Out of Memory error message. You're probably not using arrays this large, but it takes only a few multidimensional arrays to eat up all your variable memory.

A more likely possibility is that you're out of stack space. The computer reserves a section of memory called a stack, where it holds information it will need later. It's called a stack because programmers like to think of it as a stack of values and addresses, with the most recent addition on the bottom. Whenever you use a Do-Loop, a Do-While, a For-Next loop or a Gosub in a program, an entry is placed on the stack so the computer can find its way back when executing the loop or subroutine. If you're jumping in and out of loops,

or doing recursive calls of a subroutine, you can very quickly run out of stack space, resulting in an Out of Memory error. As an example, enter this one-line program and run it:

```
10 GOSUB 10
```

You'll find that almost as soon as you press return, it will print an Out of Memory error. This small program has filled the stack by repeatedly calling itself, and never reaching a Return statement, which would remove an entry from the stack.

Check your program again, and perhaps you will find that one of the above is causing the problem.

**Q** *I've been trying to join together two C-128 programs by using the command Concat "Part2" to "Part1". I've also renumbered the programs so they don't have overlapping line numbers. The problem is the C-128 keeps giving me a File Type Mismatch Error message. What am I doing wrong?*

—JAY GREEN  
PAHOKEE, FL

**A** The C-128 is already telling you what is wrong with its error message. The error message means you've told your computer to merge two program files, yet Concat is used only for data files. To join programs together, you'll need some sort of append utility program, of which there are many. Check your local user's group library or BBS—you may find one there.

**Q** *I'm trying to program sprites from Basic on my C-64, and I'm having problems getting more than one sprite to come on at a time. Assuming the variable V (for VIC II) is equal to 53248, and S is the sprite I want to turn on (0-7), I can then turn on any sprites I want by issuing POKE V + 21,21S. But when I try calling up a second sprite, the first one turns off.*

*The line I use to turn them on is: POKE V + 21,210:POKE V + 21,211. This should turn on sprites 0 and 1, but instead, sprite 0 comes on for a second, turns off and then sprite 1 is on. What's going on?*

—SHAWN ZOOWSKI  
CLEVELAND, OH

**A** The problem is created by your use of two Pokes instead of one. V + 21 is the Sprite Display Enable register, and each of the eight bits in that register determines if a sprite is on or off. If the bit equals 1, the sprite is on; if it equals 0, the sprite is off. So poking 210 turns on sprite 0, 211 turns on sprite

1, 213 sprite 2, and so on. But by poking in each value one at a time, you turn off all the preceding values. Using POKE V + 21,210 + 211 will allow you to turn on two sprites at once.

The flip side of the problem is how to turn off one sprite without turning off the others, which can be difficult if you don't know what others are on. Again, we can use the Poke statement, but this time we combine it with a Peek command. To turn off a sprite S with a value of 0-7, you'd enter POKE V + 21,(PEEK(V + 21) AND (255 - 21S)) in your program.

Take a look at what this rather cryptic line does. First, it uses the Peek command to get the contents of memory address (V + 21). Then it performs the Boolean function AND on that value, using the expression (255 - 21S) as its argument. If S = 0, then 255 - 21S = 254. ANDing the value found at V + 21 with 254 turns off bit 0 if it's on, and leaves it unaltered if it's off. Finally, this new number is poked back into memory location V + 21. In general, we can use the above procedure to selectively turn off any bit in a byte.

The earlier example above for turning on two sprites at once is fine as far as it goes. But suppose we want to turn on a specific sprite without changing any of the others. We can use a variation of the Peek and Poke statement above, only this time using the Boolean OR function to selectively set a bit: POKE V + 21,(PEEK(V + 21) OR 21S). Again, S is a value between 0 and 7 that represents the eight sprites. This Poke will turn on any of the eight sprites, and have no effect on the others. You could use these two Pokes as subroutines, which you would call to turn on and off any sprite.

**Q** *Could you give me a short and simple example for performing basic disk commands? I'm a new owner, and the 1541 and 1571 manuals are quite confusing. I have to dig hard to find what I need!*

—JERRY GOLDSTEIN  
NEW YORK, NY

**A** You're right about the manuals being confusing to the new owner. But, in general, it's a good idea for the beginner to study the manuals that come with the computer; they'll begin to make a lot more sense after a while.

In the following examples, the C-64 Basic 2.0 version is given first, followed by the C-128's Basic 7.0 syntax. I used the generic FILENAME or DISKNAME in all cases. Replace them with your own

## COMMODORE CLINIC

file or disk name when you're using the examples. All commands assume you're using drive 8, but you could change them to work on other drives if that is necessary.

To read a directory:

```
C-64:  LOAD "$",8<return>
        LIST<return>
C-128:  DIRECTORY
```

To load a program:

```
C-64:  LOAD "FILENAME",8
C-128:  DLOAD "FILENAME"
```

To save a program:

```
C-64:  SAVE "FILENAME",8
C-128:  DSAVE "FILENAME"
```

To verify a program in memory with one on disk:

```
C-64:  VERIFY"FILENAME",8
C-128:  DVERIFY"FILENAME"
```

To format a disk:

```
C-64:  OPEN 15,8,15,"N0:DISKNAME,
        XX":CLOSE 15
C-128:  HEADER "DISKNAME",XX
```

(Note: XX is any two numbers or characters to be used as a unique ID number.)

To erase a file:

```
C-64:  OPEN 15,8,15,"S0:FILENAME":
        CLOSE 15
C-128:  SCRATCH "FILENAME"
```

To rename a file:

```
C-64:  OPEN 15,8,15,"R0:NEWFILE
        NAME=0:OLDFILENAME":CLOSE 15
C-128:  RENAME "OLDFILENAME" TO
        "NEWFILENAME"
```

To initialize a drive:

```
C-64:  OPEN 15,8,15,"I0":CLOSE 15
C-128:  DCLEAR
```

To validate a disk:

```
C-64:  OPEN 15,8,15,"V0":CLOSE 15
C-128:  COLLECT
```

To switch a 1571 to 1541 mode:

```
C-64 and C-128:  OPEN 15,8,15,"U0>
        M0":CLOSE 15
```

**Q** I'm writing a program on my C-64 and would like to know how to enter the cosine formula for the trigonometric solution of a triangle when the three sides are known:  $\cos A = (b^2 + c^2 - a^2)/2bc$ . Suppose the sides  $a$ ,  $b$  and  $c$  are 8, 9 and 10, respectively.

—C. J. ERKER  
CLEVELAND, OH

**A** To employ trigonometric formulas (or any other mathematical

expressions) in a program, you'd use Basic 2.0 syntax. For the example you give, you'd enter:

```
10 A=8:B=9:C=10
20 CA=(B12+C12-A12)/(2*B*C):REM CA
    EQUALS COS OF ANGLE A
30 PRINT "COS A=";CA
```

**Q** I'd like to use my 1750 RAM expansion module with programs I'm writing to shift stored data to the REU, retrieve the data for use in the programs, put the data back into the unit so that I can use RAM memory for other work, and save the RAM data to disk for later use. Could you give me an example of how to do this?

—ROBERT E. PORTER  
CANFIELD, OH

**A** I'll give you two examples. The first is to use Commodore's official RAMDOS software, which simulates a high-speed disk drive. With a RAM drive you can quickly load and save your data between the program you're writing and the drive. You can even chain several programs together so they act as one. And it's done so fast that it's usually transparent to the user.

If you recently bought your RAM cartridge, the RAMDOS software was probably already on the disk that came with the unit. Earlier buyers of a 1700 or 1750 unit did not get it, as it wasn't completed until this year. Commodore has now released it for public use, and you'll find it on most commercial BBS networks, such as QuantumLink, GENIE and CompuServe. You'll also find it on many smaller BBSs, including RUN's own RUNning Board (603-924-9704).

The second example is to use the Basic 7.0 commands Stash, Fetch and Swap to store and recall data from your programs. I've given the commands and their parameters below.

```
FETCH,#BYTES,INTSA,EXPSA,EXPB
STASH,#BYTES,INTSA,EXPSA,EXPB
SWAP,#BYTES,INTSA,EXPSA,EXPB
#BYTES—the number of bytes to Fetch, Stash or Swap
INTSA—the starting address (0-65535) of the computer's memory
EXPSA—the starting address (0-65535) of the expansion RAM
EXPB—the memory expansion bank number (0-1 for 1700, 0-7 for 1750)
```

The commands themselves are quite easy to understand, but you must know a great deal about the 128's memory organization in order to use them. I've written a sample program that stores the 128's 40-column graphics screen in

the RAM expansion cartridge, waits for a keypress, then restores it.

```
10 GRAPHIC1,I:REM HIGH RESOLUTION
20 REM CREATE A SCREEN TO STORE
30 FOR I=1 TO 16
40 C=INT(RND(1)*16)+1
50 COLOR I,C
60 X=INT(RND(1)*320)
70 Y=INT(RND(1)*200)
80 XR=INT(RND(1)*99)+2
90 YR=XR*.769
100 CIRCLE I,X,Y,XR,YR
110 NEXT I
120 STASH 9200,7168,0,0
130 GRAPHIC I,I:REM CLEAR THE SCREEN
140 GETKEY A$:REM WAIT UNTIL A KEY IS PRESSED
150 FETCH 9200,7168,0,0:REM RESTORE THE SCREEN
160 GETKEY A$
170 GRAPHIC 0
```

This is just an example; there are other ways to write the program, depending on your needs, such as having different graphic or text screens, banks of sprites or function key definitions.

## SOFTWARE

**Q** A C-128 program I'm writing needs to know what disk drive it was loaded from and the amount of memory present. Short of the program asking the user, is there a way to tell what expansion RAM is present and which drive (8-11) should be used?

—TOM McDUNNEL  
WEST PALM BEACH, FL

**A** Yes, there is. You can tell what drive was last accessed with a Peek to \$BA, and by placing a line at the beginning of your program that Peeks that address, you'll know what drive to default to. Here's an example.

```
10 DN=PEEK(DEC("BA"))
20 BLOAD"SPRITES",B0,U(DN+0)
```

The variable DN is set to the drive last used, so if you load and run this program from any drive, the sprite data is loaded from the same drive.

As for checking for 17XX RAM expansion cartridges, again, a small routine at the beginning of your program can handle that for you. For example:

```
10 A=57094:POKE A,255:IF PEEK(A)
    <>255 THEN RX=0:GOTO 30
20 POKE A,0:A=57088:RX=128:IF
    (PEEK(A) AND 16) THEN RX=512
30 REM RX CONTAINS THE AMOUNT OF
    EXPANSION RAM PRESENT
```

## COMMODORE CLINIC

This short routine will return the total amount of expansion RAM available in the variable RX. If you have a 1700, RX=128, and if a 1750 is present, RX=512. If the system is unexpanded, the value of RX will be 0.

**Q** The Advanced OCP Art Studio saves the multicolor picture I created into one large 40-block file. How can I separate it into components that I can load directly in C-128 mode, and how can I load it once it's separated?

—STEVE DELASSUA  
FLORISSANT, MD

**A** The best method is to break down the multicolor picture file into four parts: a bitmap (8K), two color memories (1K each) and the background and border colors (two bytes). These can then be individually loaded into the areas of the C-128's memory that are necessary to display the picture. For this, I wrote two simple Basic 7.0 programs. The first (Listing 1) converts multicolor Advanced OCP Art Studio pictures into the four files. It prompts you for the name, then writes them out with the suffixes BM (bitmap), C1 (color memory 1), C2 (color memory 2) and C3 (background and border color).

The second listing, using Basic 7.0, loads and displays the picture, using the four files made with Listing 1. It can easily be used as a subroutine in your programs to display your pictures.

```
0 REM LISTING 1—CONVERT
10 PRINT "THIS CONVERTS ADVANCED
ART STUDIO"
20 PRINT "MULTICOLOR MODE
PICTURES TO FOUR FILES"
30 PRINT "THAT CAN BE LOADED FROM
BASIC 7.0"
40 PRINT
50 PRINT "NAME (MPIC SUFFIX IS NOT
NEEDED)"
60 INPUT F$
70 N$=F$+"{16 spaces}"
80 N$=LEFT$(N$,12):N$=N$+"MPIC"
90 BANK 0
100 BLOAD (N$),B0,P8192
110 BSAVE (F$+"BM"),B0,U8,P8192 TO
P16192
120 BSAVE (F$+"C1"),B0,U8,P16192 TO
P17192
130 BSAVE (F$+"C2"),B0,U8,P17208 TO
P18208
140 BSAVE (F$+"C3"),B0,U8,P18208 TO
P18210
150 BANK 15
```

```
0 REM LISTING 2 — DISPLAY
10 INPUT "PICTURE NAME";F$
```

```
20 GRAPHIC 3,1
30 POKE 1,PEEK(1) AND 254
40 BLOAD (F$+"BM"),B0,P8192
50 BLOAD (F$+"C1"),B0,P7168
60 BLOAD (F$+"C2"),B15,P55296
70 BLOAD (F$+"C3"),B13,P53280
80 GETKEY AS
90 GRAPHIC 0
```

## HARDWARE

**Q** When the C-128 came out over a year ago, I thought that besides being able to use my C-64 games and programs on it, I could also take advantage of its 80 columns for word processing and its superior graphics to play better games. I'm having no problems using my 128 word processor, but I am having trouble with games.

For example, while the 64 mode may be compatible with C-64 programs, the 1571 disk drive, apparently, is not. Many of my games do not work on the 128 in 64 mode when I use them with the 1571.

Since almost all computer outlet stores won't take back a program package that's been opened, it seems to me that it is up to the software manufacturers to alert the consumer to any quirks.

—SCOTT ANGSTREICH  
CHERRY HILL, NJ

**A** The problem you're having with software loading is related to the problem with returning it to a store once it's been opened—software piracy. In an effort to stop the wanton copying of software (particularly games), manufacturers resort to various copy-protection schemes. Many of these techniques utilize specific areas of memory or specific routines within the 1541. If everything isn't just perfect (such as a printer not connected or a cartridge not plugged in or there's a full moon), the program won't load. Even if the drive is slightly out of alignment, it might not load. So you shouldn't be surprised that some software won't work on the 128 in 64 mode when you use a 1571.

However, there are a couple of tricks you can try to get your 1571 to load 1541-formatted programs. Sometimes the 1571 is still in 1571 mode, which might be the case, for example, if you started out in 128 mode and typed in GO 64. The drive may not have been set to 1541 mode when you entered GO 64. If that's the case, type in

```
OPEN 15,8,15,"U0>M0":CLOSE 15
```

to reset the disk drive to single-sided, 1541 mode.

Another trick is to hold down the

## NOTHING LOADS YOUR PROGRAMS FASTER THAN THE QUICK BROWN BOX A NEW CONCEPT IN COMMODORE CARTRIDGES

Store up to 30 of your favorite programs in a single battery-backed cartridge for easy, instant access. Change contents as often as you wish. The QBB accepts most unprotected and "frozen" programs including the only word processor that saves your text as you type, "The Write Stuff," and the 128 terminal program, "Ultraterm III." Co-exists with GEOS® and Commodore Ram Expansion Units. Loader utilities included for C64 and C128 modes. Price: 32K \$99; 64K \$129; Utilities Disk \$6; QDisk (CP/M RAM Disk) \$10; Packages:

64K QBB + Write Stuff (C64) \$139

64K QBB + Write Stuff (128) \$144

64K QBB + Ultraterm III (128) \$139

(Add \$3 s/h & \$3 C.O.D.;  
Overseas \$5; MA res. 5%)

## Brown Boxes Inc

26 Concord Rd.  
Bedford, MA 01730  
617-275-0090,  
617-862-3675



"GOOD RELIABLE STUFF" INFO

(Jan/Feb '88)

"A LITTLE GEM" TWIN CITIES 128

(Mar/Apr '88)

"YOU'LL NEVER LOSE YOUR COOL,  
OR YOUR PROGRAMS" RUN (Nov '87)

"A WORTHY PRODUCT—LONG OVERDUE"  
AHOY (Feb '88)

Circle 420 on Reader Service card.

## BIG BLUE READER 128/64 COMMODORE <=> IBM PC File Transfer Utility

**Big Blue Reader 128/64** is ideal for those who use IBM PC compatible MS-DOS computers at work and have the Commodore 128 or 64 at home.

**Big Blue Reader 128/64** is not an IBM PC emulator, but rather it is a quick and easy to use file transfer program designed to transfer word processing, text and ASCII files between two entirely different disk formats: Commodore and IBM MS-DOS. Both C128 and C64 applications are on the same disk and requires either the 1571 or 1581 disk drive. (Transfer 160K-360K 5.25 inch & 720K 3.5 inch MS-DOS disk files.)

**Big Blue Reader 128** supports: C-128 CP/M files, 17xx RAM exp, 40 and 80 column modes.

**Big Blue Reader 64 Version 2** is 1571 and 1581 compatible and is available separately for \$29.95!

## BIG BLUE READER 128/64 \$44.95

Order by check, money order, or COD.

No credit card orders please. Foreign orders add \$4 Free shipping and handling. BBR 128/64 available as an upgrade to current users for \$18 plus original disk. CALL or WRITE for more information.



To order Call or write:  
**SOGWAP Software**

115 Belmont Road; Decatur, IN 46733  
Ph (219) 724-3900

## COMMODORE CLINIC

Commodore key when you power up or reset the computer. That will take you directly to C-64 and 1541 modes.

If you're still experiencing problems, there may be subtle differences between the 1541 and 1571 ROMs. Take the 1571 ROM to a Commodore service center for upgrading. There will be a charge, however, since it is not a free upgrade.

**Q** *I hope you can help me with a problem with my C-128. When the computer is in 80-Column RGB mode, small squares appear on the screen in column 27. While they don't interfere with anything, they do not look very good when I'm using the screen! This does not occur in 40-Column mode. Any ideas about what is wrong?*

—THOMAS FORNEY  
ANDERSON, IN

**A** It is very likely one of two things. Either you have a bad 8563 video display chip (which creates the 80-column display), or one of the two RAM chips for the 8563 (it has its own video RAM) is bad. Either way, it will require a trip to your local CBM service center. Since the RAMs are soldered to the mother board, replacing them is a job for a skilled electronic technician.

**Q** *About a year ago, I bought the new Commodore 1350 mouse, and I've been using it with programs like Pocket Writer 2 and GEOS 128. Now I hear about a new mouse called the 1351. What's the difference between them, and should I think about changing to the new version?*

—C. CALVET  
GARDEN GROVE, CA

**A** The difference is like night and day. The 1350 really isn't a mouse, although it looks like one. In reality, it's a joystick in sheep's—er, mouse's—clothing. It's only able to report to the computer movement in eight directions, just like a joystick. The 1351 mouse, on the other hand, is a "true" mouse. It reports on movement in 256 directions. (The 1351 can also be used as a standard joystick if needed.) This makes it a perfect proportional controller, and it's one of the finest peripherals ever to come out of Commodore. With software designed to use it as a mouse, its movements are smooth and fluid. I've seen it priced at under \$35—a bargain—and one I recommend.

**Q** *I recently received an SX-64 (an older, portable C-64 with built-in drive and*

*five-inch color monitor). Do the comments I read about the C-64 and the 1541 drive also apply to my SX-64? Also, I was given some older programs like The Manager and The Printed Word. How do these compare with newer programs like Pocket Filer or Pocket Writer 2?*

—L. BOWLES  
JACKSONVILLE BEACH, FL

**A** Yes, for the most part the SX and C-64/1541 are the same. The SX has a different default color on power up, though, and there may be some minor differences in the 1541 ROMs. As for the older software, if it's adequate for you, then you need nothing else. However, the newer software packages (especially Pocket Writer 2) are faster and have more features.

**Q** *While using my C-128 in C-64 mode, I formatted a disk on my 1571. The disk directory showed that I had 1328 blocks free. Does this mean my 1571 formatted both sides of the disk? I thought that happened only when you were in C-128 mode. Then, when I turned it over and tried to get a directory, it responded with Drive Not Ready. What does this mean?*

—J. P. STEVENS  
EXETER, NH

**A** It means you've formatted both sides. Evidently your 1571 was still in 1571 mode, not 1541 mode. The 1571 can be used with the C-64, although it doesn't allow the high-speed data transfer in C-64 mode. However, you have a misconception about how the 1571 double-sided drive works. It is not the same as formatting a disk with the 1541, then turning it over and formatting it on the other side, which results in essentially two separate disks, each with its own directory. To get at the data on the back of this 1541-formatted disk, you must turn it over. However, when you format with the 1571 in 1571 mode, both sides of the disk are formatted at the same time. They share the same directory, and the drive can access the data on the back without turning over the disk. When you turned the disk over, it was in the drive upside down and the directory couldn't be found, as if the disk were unformatted. This resulted in the Drive Not Ready error.

**Q** *In the November 1987 issue of RUN, you ran a review on the software Basic 8. Reviewer John Premack states, "Upgrading to 64K requires swapping your machine's 4416 or 4164 RAM chips for a pair of*

*4464s." To me, this statement is confusing because there are two rows of 4164 RAM chips, not just a pair of chips. If only a pair of chips is to be replaced, then which pair? I'm interested in buying this software, but I'd also like to upgrade my C-128 at the same time. I think your articles should be a bit more technically accurate. Thanks for your help.*

—FRANCIS J. NAPERSKY  
OWINGS, MD

**A** The 4164 RAMs John was referring to in his review of Basic 8 are those for the 8563 80-column video display chip. They're inside the small silver box on the motherboard (which also has the 8563 and Vic-II graphic chips). Please be aware that, since the chips are soldered to the motherboard, removing them and installing the two 4464 RAMs is not a job for the casual user, even if you've done soldering before. If you want the 64K for Basic 8 (and I recommend it), take the chips and sockets to a local Commodore service center. It shouldn't cost you more than \$35 to install. Doing it yourself can lead to a damaged computer. (Note: C-128Ds already have the 64K of VDC RAM as well as the latest ROM chips. I've heard of people "upgrading" their C-128Ds when there is no need to do so.)

**Q** *I always buy single-sided disks to save money. When I want a double-sided disk, I make a little notch on the left side of the disk (from the front viewpoint) with an ordinary hole-puncher. Is this safe? If not, then what can I use to make a notch?*

—WILLIAM A. ELLERBE  
DALTON, MA

**A** Using a hole puncher to notch a disk is fine, so you don't need a special gadget. But I wouldn't recommend what you are doing. When a disk is manufactured, both sides of the sheets are graded as to their quality. Only if a disk passes the standards for double-sided media is it used as a double-sided disk. The single-sided disk you're using is probably made from a batch that failed the quality standards for double-sided media, and is only reliable when the proper side is used. While you may be able to use many disks this way, sooner or later you'll lose some valuable data or programs. Since disks are now relatively inexpensive, it doesn't pay to take chances.

**Q** *I've read that the empty ROM socket in the C-128 is mapped into memory locations \$8000-\$FFFF, and that memory*



## COMMODORE CLINIC

expansion is in the same locations. I had planned on buying the Basic 8 ROM chip and the 1750 RAM expansion unit for use with Basic 8 and GEOS 128. Will the ROM interfere with the REU?

—F. KRANZ, JR.  
SEYMOUR, WI

**A** There's no conflict between the Basic 8 ROM chip and the 1750 REU. The ROM is unused and transparent unless installed into memory during system startup by holding down the control key. It works well with the 1750 REU, and, when not activated, doesn't interfere with any known software.

**Q** What is the best disk drive to use as a second drive?

—M. R. HAUGE  
SIERRA MADRE, CA

**A** It depends on your needs and your pocketbook. If you have a C-64 with a 1541 drive or a C-128 and a 1571, I would seriously consider the 1581 3 1/2-inch drive. For less than \$200, it stores over 800K and in 128 mode is very fast. If you want to stay with 5 1/4-inch drives, then a second 1541 or 1571 may be what you need. Or, if you have the money, you could get a hard drive.

**Q** I have a C-64 and plan to upgrade to the C-128D. I've heard of the 1750 RAM expansion cartridge and am wondering if it works with the 128D. Also, does it work with GEOS, perhaps by allowing more of the program to remain in memory? If I get a 1581 drive, can I transfer my protected programs to it?

—ANN BRANSTETTER  
LAUREL, MT

**A** The 1750 adds an additional 512K of memory to the C-128, which is a really impressive upgrade. It does have its limits, though, because it doesn't directly increase the memory allowed for executable programs. Instead, it acts as a storage area for programs and data. However, because of its ability to make extremely high-speed direct memory transfers (DMAs) between the C-128 and the 1750, programs and data can be loaded into the RAM expansion unit and then transferred into the computer when needed. The usefulness of this becomes evident when using GEOS, as it can practically eliminate the bothersome (and slow) disk accesses you normally encounter.

The 1581 drive can also be used with GEOS 128 (although, at this writing, not

with GEOS 64), providing very high speed with 800K storage. You can't copy most protected programs to the 1581, because it's quite different from the 1541 and 1571, but unprotected programs usually work with the 1581, and many protected programs (like GEOS 128) can use it for data storage.

**Q** Several years ago I purchased some educational software from a company called Futurehouse, Inc. Their programs need a light pen called the Edumate Light Pen, but the company went out of business before I could order one. Do you know of a source for that light pen?

—VIRGINIA HELBER  
SCOTTSDALE, AZ

**A** No, Virginia, they are no longer available. However, another pen should work quite well. Two of the best are from Inkwel Systems (PO Box 85152 MB290, San Diego, CA 92138; 619-268-8792). One of these is the model 170-C, and it retails for \$99.95. The other is the 184-C, which sells for \$59.95. Both come with some demonstration programs.

Another good light pen is available from Tech Sketch, Inc. (40 Vreeland Ave., Totowa, NJ 07512; 201-256-0013). Their LP-10 costs \$49.95 and comes with a high-resolution color drawing program.

**Q** A few months ago I bought a 1750 RAM expander for my C-128. How can I use it as a disk drive to store programs? Also, is there a program that will help me make better use of this add-on device?

—JOHN EACOTT  
WOODSTOCK, ONTARIO, CANADA

**A** You can use the 1750 (as well as the 1700 and 1764) RAM cartridge as a high-speed disk drive by using the official Commodore RAMDOS software. It recently was released into the public domain by Commodore, so it should be accompanying RAM cartridges by now. However, if you didn't get it with the cartridge when you bought it, check your local user's group libraries as well as various online services or BBSs. It's also available for downloading from the RUNning Board BBS (603-924-9704) here at RUN.

**Q** I have an SX-64 portable computer and would like to add on the new 1764 RAM expander. Is it compatible with the SX 1541 ROM? And, since the power supply on the SX is internal, how does one use the new

## COLOR RIBBONS & PAPER

COLOR RIBBONS				
RED, BLUE, GREEN, BROWN, PURPLE, YELLOW				
Ribbons	Price Each	Black	Color	Heat Transfer
Brother M1109	4.95	5.95	7.00	—
C. Itoh Prowriter Jr.	7.00	9.00	—	—
Citizen 120D/180D	5.00	6.00	7.95	—
Commodore MPS 801	4.15	4.75	5.75	—
-MPS 802/1526	6.00	6.75	—	—
-MPS 803	4.95	5.95	7.00	—
-MPS 1000	3.95	4.95	6.75	—
-MPS 1200/1250	5.00	6.00	7.95	—
-1525	6.00	8.00	—	—
Epson MX80/LX800	3.75	4.25	6.75	—
Okidata 82/92	1.75	2.25	4.50	—
Okidata 182/192	6.50	7.50	—	—
Panasonic K-XP 1080	6.75	7.75	—	—
Seikosha SP 800/1000	5.25	6.50	7.95	—
Star SG10	1.75	2.25	4.50	—
Star N10/NL10	5.00	6.00	7.95	—
Star NX1000	5.00	6.00	8.00	—
Star NX1000C—	—	10.75	—	—
4-Color				

COLOR PAPER	
BRIGHT PACK	200 Sheet/50 ea. color: Red, Blue, Green, Yellow. 9 1/2 x 11—\$10.90/pk.
PASTEL PACK	—200 Sheets/50 ea. color: Pink, Yellow, Blue, Ivory. 9 1/2 x 11—\$10.90/pk.

T-SHIRT RIBBONS (Heat Transfer)—Call For Price & Avail.
COLORS: Red, Blue, Green, Brown, Purple, Yellow, Black

COLOR DISKETTES
5 1/4" DS/DD Rainbow Pack. 10/pack—\$12.50

For ribbons & paper not listed above, call for price. Price & spec. subject to change w/o notice. Min. order \$25.00. S&H \$3.50 minimum. Visa, MC, COD

**RENCO COMPUTER SUPPLIES**  
PO BOX 475, MANTENO, IL 60950 USA  
1-800-522-6922 • (IL) 1-800-356-9981  
815-468-8081

Circle 421 on Reader Service card.

## DEALERS SELL

Selling RUN will make money for you. Consider the facts:

**Fact #1:** Selling RUN increases store traffic—our dealers tell us that RUN is the hottest-selling computer magazine on the newsstands.

**Fact #2:** There is a direct correlation between store traffic and sales— increase the number of people coming through your door and you'll increase sales.

**Fact #3:** Fact #1 + Fact #2 = INCREASED SALES, which means money for you. And that's a fact.

For information on selling RUN, call 1-800-343-0728 and speak with our Direct Sales Manager. Or write to RUN, Direct Sales Dept., 80 Elm St., Peterborough, NH 03458.

# RUN

## COMMODORE CLINIC

power supply that comes with the expander, or is the power supply built into the SX capable of handling it?

—ALAN TREMBLAY  
CHRISTOPHER LAKE, SASK., CANADA

**A** The official Commodore policy is that the 1764 is only for the C-64, and not compatible with the SX-64. However, unofficially I have been told the power supply on most SX-64s will allow you to use the RAM expander; but be warned that there is always the (small) possibility of damage. Another potential problem, and one more likely to occur with the SX-64 and 1764, comes from the VIC-II graphic chip. The VIC-II chip in some of the older SXs simply won't work with the 1764. If that happens, just have a new VIC-II chip installed in the SX.

**Q** Does it matter which serial port of my 1541 I connect my 64 to?

—JASON OEHRLE  
HIGHLAND, CA

**A** You can use either port. The reason there are two serial ports is to "daisy chain" a printer or multiple disk drives.

**Q** I have two questions. First, I've seen C-64 software available for SAT preparation. Is there any company that offers C-64-compatible software for preparing for the GRE (Graduate Record Examination)? Second, I'm thinking about getting the Educator 64 for classroom use. However, the ads don't specify what, if any, form of storage device it comes with or uses. Since it appears to be a C-64 in a CBM PET case, I was wondering if it uses a 1541 serial drive, as the C-64 does, or a 2031-style IEEE drive, like the PET?

—JACK DAVIDSON  
WABASH, IN

**A** I checked with Beth Jala, RUN's Review Editor about GRE preparation software, but she wasn't aware of any for the C-64 or C-128. As for the Educator 64, it's a C-64 in an old style PET case, but it still uses standard 1541 disk drives. Unless the ad explicitly states it includes a drive, you should assume you must buy it separately.

**Q** I own a C-128 and a 1541. I want to upgrade my drive, as well as buy a new monitor. What are the differences between the 1571 and 1581 drives, and the 1902 and 2002 monitors? I've noticed that the 1581 is less expensive than the 1571, but I'm told

it has more storage! Is it completely compatible with 64 and 128 software?

—MICHAEL GREENSTEIN  
ORANGE PARK, FL

**A** The major difference between the 1571 and 1581 is the greatly increased disk storage capacity (350K vs. 800K) of the 1581. But you should realize that there are not yet many commercial programs available on the 3¼-inch disks accommodated by the 1581. Another difference is the 1581's faster load times. Also, the 1581 can be partitioned, which lets you use subdirectories on your disks.

As for software compatibility, no, the 1581 is not completely compatible. Software with disk-based copy-protection schemes probably won't work. However, unprotected software or software that can use the 1581 as a secondary data drive should work well.

As for the 2002, it is Commodore's "universal" monitor. It has composite, RGBI and RGBA connections, allowing it to work with both the C-64 and the C-128. Commodore is no longer manufacturing the 1902, which is becoming very difficult to get.

The October 1987 issue of RUN reviews the C-128D, 1581 drive and the 2002 monitor. Refer to it for additional information.

**Q** I recently read the Basic 8 review in the November 1987 issue of RUN. The review mentioned it was possible to replace the 128's 16K 80-column VDC RAMs with 64K. Is it also possible to replace them with 256K chips? If so, this would allow more than one hi-res screen in memory at a time. Perhaps the computer could be drawing in one screen and displaying a second. Would that be possible?

—WILLIAM STOTTARD  
PASCAGOULA, MS

**A** No, it's not possible to add 256K RAM to the 128 for 80-column display purposes. The 64K (using 120 nanosecond 4464 RAM chips) is the maximum it can be expanded to. But as for the possibility of having more than one screen in memory at a time, that is already quite possible when using Basic 8. You can have up to four monochrome screens in memory simultaneously (if you have 64K of VDC RAM), and Basic 8 is designed for drawing in one screen while looking at another (a technique called double buffering). With the 64K you can also have a large variety of different-size virtual screens, as well as variable size color displays.

By the way, while older C-128's came with only 16K of VDC RAM, the new C-128D has the full 64K video memory installed at the factory.

**Q** I have a 64C, a Blue Chip BCD disk drive and a Magnavox CM 8501 Color Monitor 40. When I use Easy Script for word processing, I only get a maximum of 40 columns of text on the screen. If I were to switch monitors to a Magnavox 80-column monitor, would the display change to 80-column format? (Easy Script does have 80-column support.)

—DAVID DE ROIA  
TORRANCE, CA

**A** No. Switching to an 80-column monitor will not change your display. That's determined by the computer and its software. The C-64 uses only a 40-column text display, and changing monitors won't affect it at all. The 80-column format provided by Easy Script involves the use of scrolling the screen, but it still never displays more than 40 columns of text at a time.

There are two ways to get an 80-column display on the 64, but neither are very satisfactory solutions. One is to create a special character set using 4 x 8 pixel cells for the font, and using bit-map Graphics mode to display them. This is slower than Text mode, it's hard to read and very memory-intensive.

The other is to use an 80-column cartridge, which gives a true 80-column display. However, these are expensive and not supported by many software companies. Batteries Included (now out of business, but some of their product line is still available from Electronic Arts) at one time sold both an 80-column cartridge and a word processor, PaperClip, that supported it. The program is still available from Electronic Arts, but they don't carry any Batteries Included hardware. If you need customer assistance for a Batteries Included product, call 1-415-578-0316.

**Q** I currently own a C-64, but I'm planning on purchasing a PC10-2 (Commodore's MS-DOS compatible computer). Is there any emulator available that would allow me to run my 64 software on the PC10?

—PETE BUSHBAKER  
DEARBORN HEIGHTS, MI

**A** No, it's not possible and is extremely unlikely ever to be so. In order for any computer to emulate the 64 (or any other computer) via software, it's necessary to translate all the 64's

## COMMODORE CLINIC

instructions and abilities into those of the computer emulating the 64, in this case your PC10. Since the PC10 is a relatively slow computer, the computational overhead would be so large that even if 64 programs could be made to work, they would run at one tenth or less of normal speed. Even on a high-speed computer like the Amiga, the 64 emulators available are so slow that they are essentially useless.

The only way it could be done is to have the 64 hardware (CPU, RAM, graphic and sound chips) on a board that could be inserted into the computer (like the Amiga's MS-DOS Bridge-Board). But the cost of such a board would probably be as much or more than a 64 by itself. The only practical method is that used by the 128, which is to have the 64 hardware installed as an integral part of the computer.

It has come to my attention that a fair number of 64 users are planning on upgrading to the Amiga in the belief that they will still be able to use their 64 software, using one of the commercially available Amiga C-64 emulators.

I checked with *RUN*'s sister publication, *AmigaWorld*, about this, conferring with Guy Wright, *AmigaWorld*'s editor-in-chief (who was at one time technical editor here at *RUN* and is still a 64 user). He was at that moment writing a review on C-64 emulators for *AmigaWorld*'s February 1988 issue.

After testing both the GO 64 and C-64 Emulator programs, he came to the general conclusion that both are so slow that even the programs that will run (and most do not) are far too slow to be usable. After watching a few demonstrations, I came to exactly the same conclusion. So, better forget about using your C-64 library on the Amiga.

To be fair, however, there are two other products available that allow you to use some 64/128 hardware on an Amiga. One is the C-View cable from C-Limited. This cable will connect your Amiga 500's RGB output to a 1702 or compatible composite monitor, giving you a readable color display. Another product is Access 64, which allows you to use your 1541/1571/1581 drives as well as MPS-801 compatible printers on an Amiga, in Amiga mode. It is not an emulator, but it does have some software for transferring datafiles (like word processing files) between the Amiga and the C-64.

**Q** I need a hard drive for my C-128D. Can I use an IBM-style controller and interface for a Seagate ST-506 (5 meg), an

ST-419 (15 meg) or an ST-225 (20 meg), preferably in RLL format? Will I be able to use the hard drive with GEOS or GEOS 128? I would rather not spend the \$800 or \$900 on the system Xetec offers.

—JASON HULL  
SPRINGDALE, AR

**A** That's a great idea, but, sad to say, one that's not possible (yet). To use any hard drive, software must be provided to drive it, to perform file management of extremely large numbers of programs or datafiles, and yet still be compatible with the majority of non-copy-protected software. This is what Xetec (and others) have done. So far, no one has developed a combination interface and software for off-the-shelf hard drives. But whoever does so will likely make a lot of money!

As for using a hard drive with GEOS, current GEOS software will not work with any hard drive for a CBM computer because of copy-protection problems.

**Q** I have both the SFD-1001 and 1541 disk drives. The SFD drive is interfaced to the computer with the Skyles IEEE Flash 64 interface. My problem is finding a copy program that will let me copy programs from the 1541 to the 1001. Are there any commercial or public domain programs that do this?

—DOUGLAS BREA  
PETERBOROUGH, NH

**A** Commodore guru Jim Butterfield wrote a popular program called Copy/All 64 that will work just fine. It's available from just about every user group library, local BBS or national telecommunications network around. It's also on the 1541 Test Demo disk that came with your 1541.

But if you need to copy entire disks, using a track and sector-type copier won't work. In order to get the larger capacity of the 1001 drive, it has to use a different disk format, one that's unfortunately incompatible with the 1541. So, whole disk backups are not possible. But, as mentioned, you can copy a disk one file at a time.

Other problems will arise when you try to run some software from the SFD-1001. Anything that uses the 1541 disk ROMs for faster loading or copy protection will not work correctly on the 1001. And some software may be incompatible with your IEEE interface and not run properly. ■

Lou Wallace, the author of *Commodore Clinic* and *RUN*'s technical manager, is a prolific and expert graphics programmer.



Top-Tech International, Inc.



World's Service Perfection

## Lifetime-Computer®

Lifetime Warranty for every  
sold or serviced C-64  
(\$60.00), C-128 (\$80.00)

exclusively from

TOP TECH WORLD, INC.—  
Commodore/Amiga  
SALES AND SERVICE CENTER

Computers, Printers, Software,  
Hard-to-find Parts, Power Supplies  
Service Manuals

AMEX, DISCOVER, MASTERCARD,  
VISA

## TOP TECH WORLD

1100 S. Delaware Ave.,  
Philadelphia, PA 19147  
(215) 389-9901  
(800) 843-9901

Circle 422 on Reader Service card.

*RUN* is a publication of IDG Communications/Peterborough, a division of IDG Communications, the world's largest publisher of computer-related information. IDG Communications publishes over 90 computer publications in 33 countries. Fourteen million people read one or more of IDG Communications' publications each month. IDG Communications publications contribute to the *IDG News Service*, offering the latest domestic and international computer news. IDG Communications publications include: ARGENTINA's *Computerworld Argentina*; ASIA's *Communications World*, *Computerworld Hong Kong*, *Computerworld Malaysia*, *Computerworld Singapore*, *Computerworld Southeast Asia*, *PC Review*; AUSTRALIA's *Computerworld Australia*, *Communications World*, *Australian PC World*, *Australian Macworld*; AUSTRIA's *Computerwelt Osterreich*; BRAZIL's *Data-News*, *PC Mundo*, *Micro Mundo*; CANADA's *Computer Data*; CHILE's *Informatica*, *Computacion Personal*; DENMARK's *Computerworld Danmark*, *PC World Danmark*; FINLAND's *Tietoviikko*, *Mikro*; FRANCE's *Le Monde Informatique*, *Distributive*, *InfoPC*, *Telecom International*; GREECE's *Micro and Computer Age*; HUNGARY's *Computerworld SZT*, *PC Mikrovilag*; INDIA's *Dataquest*; ISRAEL's *People & Computers Weekly*, *People & Computers BiWeekly*; ITALY's *Computerworld Italia*; JAPAN's *Computerworld Japan*; MEXICO's *Computerworld Mexico*; THE NETHERLANDS' *Computerworld Netherlands*, *PC World Benelux*; NEW ZEALAND's *Computerworld New Zealand*; NORWAY's *Computerworld Norge*, *PC World Norge*; PEOPLE'S REPUBLIC OF CHINA's *China Computerworld*, *China Computerworld Monthly*; SAUDI ARABIA's *Arabian Computer News*; SOUTH KOREA's *Computerworld Korea*, *PC World Korea*; SPAIN's *CIMWORLD*, *Computerworld Espana*, *Commodore World*, *PC World Espana*, *Communications World*, *Informatica Industrial*; SWEDEN's *Computer Sweden*, *MikroDatorn*, *Svenska PC World*; SWITZERLAND's *Computerworld Schweiz*; UNITED KINGDOM's *Computer News*, *DEC Today*, *ICL Today*, *PC Business World*, *LOTUS*; UNITED STATES' *AmigaWorld*, *CD-ROM Review*, *CIO*, *Computer Currents*, *Computerworld*, *Computers in Science*, *Digital News*, *Federal Computer Week*, *80 Micro*, *FOCUS Publications*, *inCider*, *InfoWorld*, *Macintosh Today*, *MacWorld*, *Computer & Software News* (*Micro Marketworld/Lehar-Friedman*), *Network World*, *PC World*, *Portable Computer Review*, *Publish!*, *PC Resource*, *RUN*, *Windows*; VENEZUELA's *Computerworld Venezuela*; WEST GERMANY's *Computerwoche*, *PC Welt*, *Run*, *Information Management*, *PC Woche*.

## MAGIC

From p. 14.

Although it locks C-128 programs, it must be run in 64 mode. Furthermore, locked programs can be modified by adding lines (which will appear when listed) and deleting existing lines. You can even add dummy do-nothing lines to really confuse the peeper; just be sure to precede them with a locked line containing a GOTO xxx that prevents the dummy lines from being executed.

```
Ø REM LINE LOCK - TOM HAYS :REM*197
1Ø CK=Ø:FORL=49152 TO 49371:READD:CK=CK+D:
  POKE L,D:NEXT :REM*44
2Ø IF CK <> 34859 THENPRINT"ERROR IN DATA.
  ..":END :REM*83
3Ø INPUT"FILE TO CONVERT";P1$ :REM*161
4Ø INPUT"NEW FILE'S NAME";P2$ :REM*233
5Ø OPEN15,8,15:OPEN8,8,8,P1$+" ,P,R"
  :REM*25Ø
6Ø OPEN9,8,9,P2$+" ,P,W":SYS 49152:CLOSE8:C
  LOSE9:CLOSE15 :REM*185
7Ø DATA 162,8,32,198,255,16Ø,Ø,32,2Ø7,255,
  153,224,192,141,221,192,2ØØ :REM*135
8Ø DATA 32,2Ø7,255,153,224,192,141,222,192
  ,14Ø,22Ø,192,162,Ø,32,198,255 :REM*Ø
9Ø DATA 32,192,192,16Ø,4,169,Ø,153,224,192
  ,2ØØ,169,58,153,224,192,2ØØ :REM*1Ø1
1ØØ DATA 153,224,192,2ØØ,153,224,192,2ØØ,1
  53,224,192,162,8,32,198,255 :REM*96
11Ø DATA 169,Ø,141,223,192,32,2Ø7,255,2Ø1,
  Ø,2Ø8,3,238,223,192,32,2Ø7,255 :REM*86
12Ø DATA 2Ø1,Ø,2Ø8,1Ø,238,223,192,173,223,
  192,2Ø1,2,24Ø,75,16Ø,2,32,2Ø7 :REM*174
13Ø DATA 255,153,224,192,2ØØ,32,2Ø7,255,15
  3,224,192,16Ø,9,32,2Ø7,255,153:REM*178
14Ø DATA 224,192,24Ø,4,2ØØ,76,116,192,14Ø,
  22Ø,192,2ØØ,152,24,1Ø9,221,192:REM*197
15Ø DATA 141,221,192,169,Ø,1Ø9,222,192,141
  ,222,192,16Ø,Ø,173,221,192,153:REM*165
16Ø DATA 224,192,2ØØ,173,222,192,153,224,1
  92,162,Ø,32,198,255,32,192,192 :REM*56
17Ø DATA 76,62,192,16Ø,Ø,169,Ø,153,224,192
  ,2ØØ,153,224,192,14Ø,22Ø,192 :REM*2
18Ø DATA 32,192,192,96,238,22Ø,192,162,9,3
  2,2Ø1,255,16Ø,Ø,185,224,192,32:REM*2Ø2
19Ø DATA 21Ø,255,2ØØ,2Ø4,22Ø,192,2Ø8,244,1
  62,3,32,2Ø1,255,96 :REM*121
```

—TOM HAYS, GORE, VA

## 5. DISK DRIVES, 1541/1571/1581

### MORE FILENAME EXAMINATIONS

Many newcomers (and a few old-timers) to Commodore computers probably don't realize how versatile the LOAD"\$",8 statement really is. Did you know, for example, that appending a colon and ten question marks loads only those filenames made up of ten characters? You type in the command as:

```
LOAD"$:?????????" ,8
```

Use eleven question marks for a list of eleven-character filenames, and so on.

Commodore 128 owners can use:

```
DIRECTORYU8,"?????????"
```

to display those files with ten characters. Finally, you can see a particular program on disk with the command:

```
LOAD"$:filename",8
```

or, in C-128 mode:

```
DIRECTORYU8,"filename".
```

—TIM WALSH, RUN STAFF

### 1581 FILENAME EXAMINATION

When you load a program file using an asterisk wild card from either a 1541 or 1571 disk drive, filename information to the right of the asterisk is ignored, such as the N in the filename, R\*N.

However, the 3 1/2-inch 1581 disk drive interprets this type of filename differently. It does not ignore characters following the asterisk wild card, but instead loads the first file beginning with an R and ending with an N. The 1581 also works this way with C-128 disk commands such as BLoad and DLoad.

—TIMOTHY J. SLATE, BRATTLEBORO, VT

### SPLAT FILE RESCUE

Nothing's worse than saving a program or file to disk, only to discover it's been converted into a splat, or open, file (marked by an asterisk) and will not load or read properly. After saving some valuable data to a sequential file, I was rudely informed by the disk directory that it had mysteriously evolved into a splat file.

In desperation, I used the Copy command to make a copy of the splat file. Eureka! My data was saved and so was my day! The only word of caution is to be sure there's sufficient space left on your disk before using this trick. Here's the Direct mode syntax for using the Copy command:

```
C-64: OPEN 15,8,15,"CØ:NEWFILE = OLDFILE":CLOSE 15
```

```
C-128: COPY "OLDFILE" TO "NEWFILE"
```

—BILL ROEPKA, TULSA, OK

### SPLAT FILE RESTORATION

Commodore computers offer a rarely used and little-known M command that can also be used to restore the contents of a splat file. Here's the typical syntax:

```
OPEN 8,8,8,"Ø:FILENAME,S,M"
```

Unfortunately, not all splat files are restorable. If you find the file cannot be restored using the M command, or if you want to delete the splat file after examining its contents, don't risk corrupting your disk by trying to scratch it. Instead, remove the offending file with the Validate command as follows:

```
Basic 2.0: OPEN 15,8,15,"V":CLOSE 15
```

```
Basic 7.0: COLLECT
```

—JONATHAN PENTON, NORCROSS, GA

### 1541 DIRECTORY HIDER

Keeping your disk directory hidden away from prying eyes is easy. Just type in my program, 1541 Directory Hider, save

## MAGIC

it to disk, then run it. Two prompts appear, asking you to hide or restore a directory. Select Hide, place the disk (notched) containing the directory you want to hide in the drive and let the program do its work.

To restore the directory, run the program again. Place a disk containing a hidden directory in the drive and select the Restore option.

```
Ø REM DIRECTORY HIDER - STEVE JOHNSON
:REM*21Ø
1Ø PRINT"{SHFT CLR}":PRINTTAB(12)"MENU"
:REM*14
2Ø PRINTTAB(12)"----"
:REM*91
3Ø PRINTTAB(6)"(H)IDE DIRECTORY"
:REM*117
4Ø PRINTTAB(6)"(R)ECOVER DIRECTORY"
:REM*232
5Ø GETB$:IFB$=""THEN5Ø
:REM*5
6Ø IFB$="R"THENPRINT"{SHFT CLR}":GOTO16Ø
:REM*151
7Ø IFB$="H"THEN9Ø
:REM*25Ø
8Ø GOTO5Ø
:REM*21Ø
9Ø PRINT"{SHFT CLR}1541 DIRECTORY HIDER"
:REM*3Ø
1ØØ PRINT"THIS HIDES YOUR DIRECTORY ONLY A
FTER"
:REM*25Ø
11Ø PRINT"A DIRECTORY HAS BEEN LOADED FROM
BASIC."
:REM*19Ø
12Ø PRINT"IT WON'T WORK IF DIRECTORY IS LO
ADED"
:REM*69
13Ø PRINT"USING OPEN STATEMENTS."
:REM*145
14Ø PRINT"MAKE A BACKUP FIRST."
:REM*31
15Ø PRINT"{2Ø COMD Ts}"
:REM*12
16Ø PRINT"INSERT DISK IN DRIVE 8"
:REM*Ø
17Ø PRINT"{2 CRSR DNs)PRESS [RETURN]
:REM*254
18Ø FORX=1TO7:S$=S$+CHR$(Ø):NEXTX
:REM*88
19Ø GETA$:IFA$<>CHR$(13)THEN19Ø
:REM*216
2ØØ IFB$="R"THENGOSUB27Ø
:REM*84
21Ø OPEN15,8,15:OPEN5,8,5,"#"
:REM*22
22Ø PRINT#15,"U1";5;Ø;18;Ø
:REM*215
23Ø PRINT#15,"B-P";5;164
:REM*5Ø
24Ø PRINT#5,S$;:PRINT#15,"U2";5;Ø;18;Ø
:REM*235
25Ø CLOSE5:INPUT#15,E,E$,T,S:CLOSE15
:REM*1Ø2
26Ø PRINT"STATUS ->"E;E$;T;S:OPEN15,8,15,"
I":CLOSE15:END
:REM*48
27Ø S$=""S$=CHR$(16Ø)+"2A":FORX=1TO4:S$=S
$+CHR$(16Ø):NEXTX:RETURN
:REM*131
```

—STEVE JOHNSON, SUTTER CREEK, CA

### 1541C RATTLE

Caution: The following trick is for experienced disk drive users, not novices. It may void your disk drive's warranty and can cause damage if performed incorrectly. If you're unsure how to perform this modification, take your 1541C to an authorized Commodore service center.

If you own a 1541C (the newest version) disk drive with the lever-type door catch, and it makes a rattling noise as though it's about to format a disk, you can cure it in short order. The jumper wire to the track-1 sensor was left uncut, so the sensor is turned off, but the 1541's DOS cannot detect

its status. Cut jumper J-3, seal the end of the wire and the problem disappears.

—NEW MEXICO USER'S GROUP NEWSLETTER

### DISK REVIVING MADE SIMPLE

Did you ever receive a slightly crumpled computer disk in the mail and discover it would not load on your computer system? Later, after returning it to the manufacturer, you're informed that it worked perfectly on their equipment. The problem may not be with your disk drive, but quite possibly caused by the hole in the damaged disk slipping and sliding in the drive.

Commercial software manufacturers use powerful disk drives that could double as farm equipment, so the disk may not slip on its hub in their drives, while it does in yours.

Here's a little trick to revive crumpled disks. Lay the disk (in its jacket, of course) on a firm, flat surface and gently rub the four edges of the disk with the smooth side of a pen or pencil. More often than not, you'll have the disk turning freely and in readable condition.

—MATT HAUG, STEINAUER, NE

### DISK DRIVE IDENTIFIER

Commodore 64 programmers who write disk backup or directory programs, or any others that need to detect which type of drive(s) is present, will appreciate my program, Disk Drive Identifier.

As a stand-alone program, this disk utility detects all active drives connected to your computer and prints the number of drives, their device numbers and model numbers. Moreover, this program works with most models of the Commodore 1541, 1571, 1581 and even some third-party 5 1/4-inch disk drives. Programmers are likely to find many other uses for this routine in their own programs.

```
Ø REM C-64 DISK DRIVE IDENTIFIER - CHRIS
HAND
:REM*2Ø8
1Ø DR=Ø:FORX= 8 TO11:POKE 144,Ø
:REM*38
2Ø OPEN1,X,15
:REM*1Ø4
3Ø POKE 144,Ø:POKE 78Ø,X:SYS 65457:REM*14Ø
4Ø POKE 78Ø,111:SYS 65427:SYS 65448:REM*8Ø
5Ø Z= ST:SYS 65454:IF Z<Ø THEN9Ø
:REM*1Ø9
6Ø CLOSE1:OPEN1,X,15,"UI"
:REM*218
7Ø INPUT#1,EN$,EM$,ET$,ES$
:REM*192
8Ø U(DR)=X:U$(DR)=RIGHT$(EM$,4):DR=DR+1
:REM*22Ø
9Ø CLOSE1:NEXT:DR=DR-1
:REM*2Ø2
1ØØ PRINT"DRIVES ONLINE:"
:REM*111
11Ø IF DR<Ø THENPRINT"NONE":END
:REM*4
12Ø FOR X=Ø TO DR:PRINT"{3 SPACES}";U$(X),
U(X)
:REM*174
13Ø NEXT
:REM*5
```

—CHRIS HAND, CHURCHVILLE, NY

### DE-SELECTING C-128 AUTO-BOOT

While the C-128's auto-boot file option is a handy feature, having it installed on your often-used disks can also lead to aggravation. Resetting the computer or simply turning it on makes the system try to load and run an auto-boot file on a disk in the drive. Furthermore, some programs autoboot when you exit them. An example of the latter occurs if you've ▶

installed an auto-boot on RUN Script 128. Exiting the program with the disk in the drive causes RUN Script to re-boot.

Here's a simple solution to avoid booting an unwanted autoboot file: Don't reset the computer or turn it on with the autoboot disk in the drive. But if you do, press the run-stop/restore key combination the instant the Commodore message screen appears. You should get a clear screen and a flashing cursor. Then, in Direct mode, enter DCLOSE. Failing to do so gives you a "FILE OPEN" message the first time you press F3 for a directory.

—ROBERT V. TAYLOR, LITTLE ROCK, AR

## DIRECTORY, PLEASE

When your C-64 has a Basic program in memory and you need to examine the disk directory without disturbing the program, enter the following command in Direct mode:

```
POKE 44, PEEK(46) + 1 LOAD "$", 8 LIST
```

To remove the directory from memory and retrieve your Basic program, enter:

```
POKE 46, PEEK(44) - 1:POKE 44, 8
```

Write these commands down and keep them close to your computer for quick reference.

—HELEN ROTH, LOS ANGELES, CA

## FILE REPOSITIONER

Long-time Commodore users agree that it makes good programming sense to place your most frequently used files near the beginning of your disk directory. It not only makes loading easier, but also significantly faster on disks containing many files.

Unfortunately, repositioning files on a disk directory is not an easy process without a disk utility. If a file named Ants needs to occupy the first file position in the directory, but that slot is already occupied by a file called Bees, you're in for a little work. I wrote File Repositioner to make the process easier. The program runs in either 64 or 128 mode and is a handy utility that can be used with all of your disks.

After running the program, you're prompted to insert the disk containing the files to be repositioned. Make sure a write-protect tab is not on that disk. Next, you're prompted to enter the name of the file that you wish to switch positions with the second file. That's all there is to it, because the program takes care of all the rest!

```
Ø REM DIRECTORY RE-ARRANGER - DEAN YAMADA
  JR. :REM*48
1Ø PRINTCHR$(147):X=15:OPENX,8,X :REM*9
2Ø PRINT"INSERT DISK CONTAINING FILES TO S
  WAP" :REM*32
3Ø PRINT"PRESS 'Q' TO QUIT" :REM*158
4Ø PRINT"ENTER NAME OF FIRST FILE":INPUT$
  :REM*251
5Ø IF N$="Q" THEN END :REM*21Ø
6Ø PRINT#X,"CØ:MOVE=Ø:";N$ :REM*212
7Ø PRINT#X,"SØ:";N$ :REM*164
8Ø PRINT#X,"RØ:";N$="Ø:MOVE":PRINT:REM*115
9Ø PRINT"ENTER NAME OF 2ND FILE":INPUT A$
  :REM*217
1ØØ PRINT#X,"CØ:MOVE=Ø:";A$ :REM*25
11Ø PRINT#X,"SØ:";A$ :REM*94
12Ø PRINT#X,"RØ:";A$="Ø:MOVE":CLOSEX
```

```
13Ø PRINT"ALL DONE!"
```

```
:REM*19
```

```
:REM*71
```

—DEAN M. YAMADA, TEMPLE, TX

## 6. GENERAL HINTS AND TIPS

### HELPFUL PRINTER HINTS

If you want to explore your printer's capabilities without the hindrance of your printer interface's limitations, then you should try printing with your interface set in Transparent mode. While this mode makes your printer a little less software-friendly, it allows a greater choice of densities when printing graphics and also prevents unwanted Commodore codes from interfering with your printer's output.

Transparent mode is accomplished on most printer interfaces by setting a DIP switch or two or by using the command:

```
OPEN 4,4,4:PRINT #4:CLOSE 4
```

or

```
OPEN 4,4,5:PRINT #4:CLOSE 4
```

If you're using a software package such as GEOS, you won't be able to use a Commodore-compatible printer driver; instead you'll have to activate your printer's driver or one that closely resembles it.

Another handy printer trick applies to anyone who experiences problems printing quotation marks in word processor documents. If you open a quotation mark, then experience a loss of left-margin space or other odd occurrences when printing, your printer may be receiving an extra code or two, triggered by the presence of the quotation mark.

You can fix this problem by getting out your printer manual and referencing the section on redefining characters. Redefine your printer's @ or £ sign into a set of quotation marks. Then, instead of using quotation marks in your document, substitute the @ or £ sign. When you print the document, the printer will print the @ or £ sign as a quotation mark.

—KEITH SILLS, NEW YORK, NY

### MAGNIFYING THE TYPE

People, like myself, who are enjoying computing in their golden years may suffer from imperfect eyesight. I discovered a \$4 magnifying bar at a drug store and have been happily using it ever since for typing in program listings from RUN. I place the bar over each line of the listing and it doubles the size of the print, making it much easier to read.

—CLYDE R. LOVELACE, KILAUEA, HI

### FIXING TEMPERAMENTAL C-128S

Nearly every C-128 owner eventually encounters what I call the "Some-Keys-Ain't-Workin' Syndrome." This ailment, which only seems to affect older, flat 128s, can strike at any time. When it does, up to half the keys on the keyboard refuse to work, regardless of whether it's in 64 or 128 mode. It's usually caused by a loose keyboard ground wire.

If the computer's 90-day warranty has expired, don't pack it up for a ride to your nearest authorized Commodore service center. Instead, try this money-saving, five-minute trick that a Commodore technician showed me several years ago:

First, unplug all the cables from the computer. Next, flip

## MAGIC

it upside down and remove the Phillips screws holding the top and bottom cases together. Carefully separate the cases and unplug the LED power light connector.

At this point, you'll notice a short, thick, braided ground wire holding the keyboard to the green motherboard. That, fellow C-128ers, is the culprit. Loosen the ground wire at the motherboard with a Phillips screwdriver and re-tighten it to give the keyboard a better ground.

Now re-assemble the computer by first plugging the power light connector back in, then press the cases together, tighten the screws and plug all the cables back into the computer.

Your entire keyboard should work fine now. I've performed this quick repair on half-a-dozen or more C-128s, and only one of them failed to respond. Sometimes a trip to the service center for more extensive repairs is in order, but you should always try tightening the ground wire first.

—TIM WALSH, RUN STAFF

## WHY AN AMIGA MONITOR?

Did you know that the original Amiga video monitor manufactured for several years by Commodore exclusively for the Amiga 1000 also makes a top-notch C-128 RGB/composite monitor? It sports a color RGB 80-column video port plus RCA plug-compatible chroma, luma and sound ports for your C-128's 40-Column mode.

While this monitor is no longer manufactured, my advice is to get one the way I did—by keeping an eye on the computer classifieds for a used Amiga and color monitor. Some Amiga owners might even be willing to sell the monitor separately from the computer. If so, you'd end up with a beautiful, reliable RGB color monitor for your C-128 at a fraction of the cost of a comparable new monitor.

—NINA OLSON, SAN ANTONIO, TX

## MAILING LABEL OR DISK LABEL

Having trouble fitting a dozen or more filenames on disk labels? Can't seem to get the little rascals to stick to the disk? If so, make a quick trip to your local stationery or department store and get a small box of 1.5x3-inch mailing labels. Not only do they adhere more securely to disks, but they also offer appreciably more space for filenames than conventional disk labels.

—PHILIP KINNEY, COLORADO SPRINGS, CO

## MORE ON MAILING LABELS

Mailing labels are great time-savers for anyone who uses automatic teller machines (ATMs). Print a batch of mailing labels containing your name, account number and other ATM-required information. Ask for a few extra ATM envelopes at your bank and stick the labels to the envelopes. The result is that you waste no more time filling out envelopes at the ATM!

Another mailing label trick applies to anyone who prints them. Beware: Sooner or later a label might peel off the roll and lodge beneath your printer's platen (roller). Don't panic—you won't need to disassemble the printer. Just get a bottle of rubber cement solvent and dribble a teaspoon or so down beneath the platen, and the label should loosen and roll out. Have a pair of tweezers handy if the jammed label is stubborn. Clean the platen thoroughly with more solvent to remove any residue before continuing your printing chores.

—LONNIE BROWN, LAKELAND, FL ■



## NEW ULTRABYTE V6.0 DISK NIBBLER

### NIBBLE COPIER WITH 305 PARAMETERS FOR COMMODORE 64 AND 128

- Copies most protected disks in 2 minutes without need for parameters including rapid-locked
- 305 parameters to make unprotected copies of recent programs including VMAX protection. 100 more than V5.0. Send stamped envelope for list
- Copies up to 40 tracks using 1 or 2 1541 or 1571 drives. Copies both sides on 1571
- Copies itself (for this reason, no refunds given)

### V6.0 \$29.95 PLUS \$4.00 SHIPPING

- |  |         |  |
|--|---------|--|
| 1. Disk Surgeon V2.0 -- new disk utility ..... | \$10.00 |  |
| 2. Ultramail -- mail list and labels .....     | \$10.00 | } both for<br>Handy-Capper -- race handicapper ..... |
| 3. McMurphy's Mansion -- text adventure .....  |         |  |
| 4. 150 older parameters and file copier .....  | \$10.00 |  |

Add \$4.00 shipping (covers up to 5 items)

Mastercard, Visa, Check or M.O., Calif. add 6.5% (\$1.85) sales tax. Foreign orders/COD add \$2.00. Payment must be in U.S. funds

UPDATES - V6.0 is same as V5.0 but with 100 new parameters. Return original Ultrabyte parameter disk with \$15.00 plus \$4.00 shipping. Foreign add \$2.00. No exceptions.

To order, write or call 24 hr. order line. For info, write.

**ULTRABYTE (818) 796-0576**  
P.O. Box 789 LaCanada, CA 91011 USA

Circle 415 on Reader Service card.

AMIGA	
AMIGA 500 COMPUTER .....	\$ 545.95
AMIGA 2000 COMPUTER .....	\$1445.95
AMIGA 501 512K RAM .....	\$ 149.95
AMIGA 2052 2MEG RAM .....	\$ 399.95
AMIGA 2088D BRIDGECARD .....	\$ 499.95
AMIGA 1010 EXTERNAL DRIVE .....	\$ 199.00
AMIGA 2090 HARD DRIVE CONTROLLER .....	\$ 319.00

## commodore

64-C COMPUTER .....	\$159.95	LT KERNAL 20M 64 .....	\$ 799.95
128D COMPUTER .....	\$449.95	LT KERNAL 40M 64 .....	\$1295.95
1764 256K RAM .....	\$118.95	LT KERNAL 20M 128 .....	\$ 849.95
1750 512K RAM .....	\$149.95	LT KERNAL 40M 128 .....	\$1345.95
1571 DRIVE .....	\$221.95	AVATEX 1200E MODEM .....	\$ 79.95
1581 DRIVE .....	\$179.95	AVATEX 1200HC MODEM .....	\$ 99.95
1541-II DRIVE .....	\$175.95	AVATEX 2400 MODEM .....	\$ 179.95
1670 MODEM .....	\$ 79.95	OMNITRONIX INTER .....	\$ 45.95
1680 MODEM .....	\$ 99.00	SKYLES QUICKSILVER .....	\$ 119.95
SUPRA 2400 MODEM .....	\$149.95	SFD 1001 .....	\$ 169.95
C-NET 64 BBS .....	\$ 84.95	C-NET 128 BBS .....	\$ 89.95



### SOFTECH COMPUTER SYSTEMS

Post Office Box 23397

Lexington, KY 40523

(606) 268-2283

(800)/992-SCSI (Orders)

No surcharge for MC/VISA

Sorry, no walk-in customers. All returns must have an RMA#. Merchandise found defective will be repaired or replaced. We do not offer refunds for defective products or for products that do not perform satisfactorily. Prices are subject to change without notice.

Commodore is a registered trademark of Commodore Business Machines, Inc. AMIGA is a registered trademark of Commodore-Amiga Inc.

Circle 417 on Reader Service card.

# RUN CLASS ADS

**COMPUTER REPAIR • FAST TURNAROUND**  
**C-64-\$45.00 1541-\$69.00**  
 \$CALL Amiga, PC10, C-128, 1571, MSD  
 Monitors, Printers, etc.  
 Includes all parts and labor  
 Commodore Authorized Service Center  
**T.C. SERVICE**  
 PO Box 1224, Hwy. 36W, Ironman Road, Hartselle, AL 35640  
 (205) 773-1322

**Repair Disk Drives with Physical Exam**  
**Surgically Precise!**

- Illustrated manual
- True digital alignment disk
- No special scopes or tools needed
- Used by many repair shops and owners

**Specify Commodore Disk Drive, 1541, 1571, etc**  
 Cardinal Software, 14840 Build America Dr., Woodbridge, VA 22191,  
 Info: (703) 491-6494  
**\$39.95 ea.** +\$3.50 S&H **800 762-5645**

**SAVE YOUR FAMILY!**  
 Genealogy software with features to fit every budget and requirement: LINEAGES/Starter, LINEAGES/Standard, LINEAGES/Advanced, and our most comprehensive FAMILY ROOTS. Prices \$29 & up. Data is compatible between systems and you may upgrade from one to another. Call for FREE information. Satisfaction guaranteed.  
**QUINSEPT, INC.**  
 PO Box 216, Lexington, MA 02173  
 1-800-637-ROOT 617-641-2930

**Introduce Your C-64 to the Rest of the World**

- 1. PRINTER UTILITY PACKAGE:** Add any or all of these printed programs to your own programs, use the plans to build a cable, and interface directly to any parallel printer without any extra electronics! \$12.50 (add \$7.50 for all programs on disk).
- 2. BUILD YOUR OWN UNIVERSAL RS-232 INTERFACE:** Includes plans, schematics, several versions, assembly options, parts sources! \$7.50
- 3. BUILD YOUR OWN STEPPER MOTOR CONTROLLER:** \$5.00
- 4. MORE UTILITIES:** Send a #10 SASE for product information sheets on new items including Interrupt Utilities, Universal Floppy Disk Controller.

Check or money order to: **SOLUTIONS**, PO Box 19774, Seattle, WA 98110

**ATTENTION HORSERACING FANS!**  
 COMPUTER-AIDED HANDICAPPING makes the difference with simple-to-use, easy-to-understand racing programs from COM-CAP! Written for all levels from beginner to expert!

**NO GAMES—NO "SYSTEMS"**  
 Speed Ratings-Pace-Trainer/jockey database-tote board aids and more!  
 C-64/128, IBM and all Sharp pocket computers

"The leading source is... COM-CAP"  
 —James Quinn, High-Tech Handicapping

Write for your FREE copy of *Understanding Computer Handicapping* AND a free catalog of programs by George Kaywood!  
**COM-CAP, PO Box 34575, Omaha, NE 68134-0575**

**SURVEYOR I**  
 Triangulation: Horizontal & Vertical Curves, Radial Stationing and Printout for both curves, Misc., Volumes in Cu. Ft., Cu. Yds. & Gals.

**SURVEYOR II\***  
 Traverse Closure: By Compass Rule & Coordinate Method, Average End Area & Single End Area Computations, Dist, Bet, Coord, w/bearings: Angle Bet, Bearings: Etc.

**R&R HISSA 9500 S.W. 51 TERR, MIAMI, FL 33165**  
 C128 or C64 \$29.95 ea.; PC \$34.95 ea.; Postpaid; Fla. Res. Add 6% Tax  
 Color/Graphics Card required for PC; \*Printer required for Surveyor II

**TRY BEFORE YOU BUY!** Yes We Accept:   

*Best selling games, utilities, educational, and classics plus new releases!*

- 100's of titles
- Low prices
- Same day shipping
- Free brochure

**RENT-A-DISC**  
 Frederick Bldg. #223  
 Huntington, WV 25701  
 (304) 529-3232

**WIN LOTTO MILLIONS!!!**  
**NEW RELEASE! LOTTO PICKER™ PLUS v2.1**

Lotto Picker™ Plus stores winning Lotto 6/7, Keno 10/11, & Pick 3/4 numbers & uses multiple statistical analysis (hot, cold, & unbiased numbers) to wheel what might be your million dollar ticket! Guaranteed to work for all Lotto-style games worldwide. Easy-to-use, fully documented, and not copy protected. Includes a database editor, programmable games, and much, much more! Never obsolete—Pays for itself! **\$34.95** (Plus \$5.55 S&H). **ORDERS: 1-800-634-5463 ext. 293. GE RIDGE SERVICES, 170 Broadway, Suite 201-RS, New York, NY 10038.** NY residents add sales tax. For IBM-PC & compatibles, PS/2, C64/128 & Apple II. Inquiries: 718-317-1961. IBM-Apple 3.5 inch—add \$10.00.



**NOW AVAILABLE FOR THE AMIGA!**

The MicroFlyte JOYSTICK, the only fully proportional continuously variable joystick control for Flight Simulator II  
 "...it transforms an excellent program into a truly realistic flight simulation system" B.A.C.E.

**MICROCUBE PRODUCTS**

Commodore 64/128

- MicroFlyte ATC Joystick . . . . . \$59.95
- Test/Calibration Disk: A diagnostic tool for your joystick . . . . . \$ 4.95

Amiga

- MicroFlyte Joystick—Plugs into the mouse port & works with most software . . . . . \$119.95
- Analog Joystick. . . . . \$ 59.95

Include \$4.00 shipping of joystick orders. FSII is a trademark of subLOGIC Corp.



**MICROCUBE CORP., PO Box 488, Leesburg, VA 22075 (703) 777-7157**

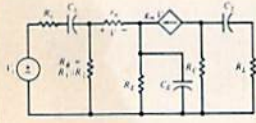
**MAKE FACES AT YOUR C-64**  
**SCAN-ON-PRINTER!**  
**"UN-VIDEO" CHOICE**

SYSTEM \$79.95 7-SIZE PROG \$39.95  
 BOTH JUST \$99.95! DOCS + SAMPLES \$1

SCAN DEMO DISK \$5  
 COD's (809)-829-4220 [2-9PM EST]  
 KALTEK/GREAT HEAD  
 ADJUNTAS PR 00601



**ELECTRONICS AC/DC CIRCUIT ANALYSIS PROGRAM** \$29.95 Disk, Tape  
 Computer Heroes    
 PO Box 79A  
 Farmington, CT 06034 C-64, C-128, IBM PC  
 Orders only 1-800-622-4070



Program computes general numeric solution to electronic circuit of up to 40 nodes and 63 branches. Branches may contain resistors, capacitors, inductors, current sources, voltage sources or 4 types of controlled sources. Computer displays node voltages, branch voltages, currents, powers and power factors. Step function of branch parameters or frequency with graphic display of results. Menu controlled and user friendly.

**FINALLY! A Music Program that is Easy to Use and Powerful**  
**The MAESTRO! for the C-64 and C-128 (64 mode)**

Easiest and fastest music entry and playback. Powerful—can accommodate almost all popular and classical music. Add feeling by conducting music using keyboard and up to eleven designated changes each, in tempo and volume. Orchestrate up to four different sounds for each voice. Cut and Paste. Transpose. Play part of a song. Play program of up to 20 songs in any order. Excellent for learning individual vocal parts. Includes 24 sample songs, and Sound Designer program. Joystick needed for music entry only. 5 1/4 in. diskette and manual—\$24.95 + \$3 shipping. CA residents add tax.

**Zwetzg Associates, Dept. S-R, 5932 Bruns Ct., Oakland, CA 94611**



## RUN Class Ads

RUN Class Ads were specifically designed to provide the effectiveness of display advertising at the cost of classified advertising. This opportunity gives the Class Ad buyer the lowest cost available to reach RUN's highly qualified circulation of exclusive Commodore 64 & 128 owners. Need help in designing your Class Ad, questions about rates, frequency or size? Call **HEATHER PAQUETTE** at 1-800-441-4403 or 603-924-9471. We accept checks, money orders, Master Card or VISA.

## QUALITY SOFTWARE

**NEW Products:** Outlining; Accounting; More  
**FREE Catalog** of these and other products  
**DISCOUNT Prices** by ordering from publisher

Call 415/563-0660 or Write:

# XYTEC

1924 Divisadero, San Francisco, CA 94115

## FREE CATALOG

Discount software for your home computer.

Apple, Atari, Commodore, IBM. . .

WMJ Data Systems-R

4 Butterfly Drive

Hauppauge, NY 11788

1-800-962-1988 Ext. 122 (516) 543-5252

## Commodore Amiga Service Center

C-64 Repair	C-128 . . . \$64.95
\$39.95	1541 . . . . 25.00 (alignment)
includes parts/labor	1571 . . . . 25.00 (alignment)
	SX-64 . . . . 59.95

CALL for Commodore Chips & Power Supplies  
at low prices.

**A&M Computer Repair**

20 Guernsey Drive, New Windsor, New York 12550  
(914) 562-7271

## BASIC GAME DESIGN Flashy Tricks of the Trade

Clean, fast BASIC for Scrolls, Animation, 3D Color Graphics, Sound and Music. Great NEW (c) GAMES fully explained. Any disk \$15 ppd. Any 2 \$25 ppd.

- 1) 10 Games of Logic—+ bonus ACTION Game, OIL WAR, + DEMOS.
- 2) 10 Games of Action—+ bonus LOGIC Game, DUFFY'S DRAWERS, + DEMOS.
- 3) Music—Easy 3-Part HARMONY, with Tremolo, Phase, Various Voices, etc. OVERTURE—play a one note melody and get Harmony. Save/load tunes to disk and EASILY add Rich MUSIC to your Own Programs! OMNIVOX—Realtime Harmony at the touch of a key! ANDROID SYMPATHY ORCHESTRA—C-64 Music in Harmony.
- 4) Advanced—WIZARD'S TOWER 1-4 Thieves of Magic face Wizard the Mad, Humor-Treachery. EMPIRE STAR a battle in 3D CubeSpace, 1-2 players-Logic. MY DEAR DR. WATSON—at last the rest of Holmes' Adventures can be told, Text-Humor. Includes AUTORUN, UNL-IST, PIX, SIDLAB and More! NEW! HOT! ROXTAR (ML)—Turns C-64 into MAXI ROCK CHORD ORGAN. \$20 ppd.

RKDO Graphics, Rte. 1 Box 199A, Stanley, WI 54788

## Commodore Service

**C-64**  
**1541 \$25.00**  
**C-128 \*Plus Parts**  
**1571**

AUTHORIZED COMMODORE  
SERVICE CENTER  
\*\*\*\*\*  
WE DO WARRANTY REPAIRS  
CALL FOR DETAILS

We also service Amiga & other Commodore equipment, Leading Edge, Star Micronics, IBM PC & XT, and Blue Chip. Call for rates.

**90-DAY  
WARRANTY**  
ON ALL REPAIRS

**TYCOM, INC.**  
503 East St.  
Pittsfield, MA 01201

**(413) 442-9771**



## RUN'S CHECKSUM

TYPE IN RUN'S CHECKSUM, which serves for both the C-64 and for the C-128 in either 40- or 80-Column mode, and save it to disk before running. When typing in a program from RUN, first load and run RUN'S Checksum. The screen will display a SYS number that deactivates and reactivates the Checksum. Always disable RUN'S Checksum before attempting to run another program. Note: You can abbreviate Basic keywords; spaces affect the checksum only when within quotes; and the order of characters affects the checksum.

With this new version, when you press return after typing in a program line, a one-, two-, or three-digit number from 0 to 255 appears in the home position. If this number matches the checksum value in the program listing, the line is correct. If the number that appears *doesn't* match the checksum value, compare the line with the magazine listing to find your error. Then move the cursor back up to the line and make your corrections. Now, after you press return, the correct checksum value should appear. Continue entering the listing until all the lines have been correctly typed. Then deactivate RUN'S Checksum, using the SYS number. Save the finished program.

All the graphics and control characters in the listings in RUN have been translated into understandable key combinations. They are the instructions you see inside the curly braces. For example, {SHIFT L} means you hold down the shift key while you press the L key. You do *not* type in the curly braces. What appears on the screen will look quite different from what is designated inside the braces. Here are some more examples:

- {22 SPACES}—press the space bar 22 times
- {SHIFT CLR}—hold down the shift key and press the clr-home key
- {2 CRSR DNs}—press the cursor-down key twice
- {CTRL 1}—hold down the control key and press the 1 key
- {CMD T}—hold down the Commodore logo key and press the T key
- {FUNCT 1}—press the F1 key
- {5 LB.s}—press the British pound key (*not* #) five times **R**

**Listing 1. RUN'S Checksum program. This program is available on RUN'S BBS for users to download.**

```

10 REM RUN'S CHECKSUM 64/128 - BOB KODADEK
20 MO=128:SA=3328:IF PEEK(40960)THEN MO=64:SA=4
  9152
30 FOR I=0TO169:READB:CK=CK+B:POKE SA+I,B:NEXT
40 IF CK<>20651 THENPRINT"DATA ERROR!":END
50 POKESA+110,240:POKESA+111,38:POKESA+140,234
60 PRINTCHR$(147)STR$(MO)" RUN CHECKSUM":PRINT
70 PRINT"TO TOGGLE ON OR OFF, SYS"SA:IF MO=128
  THEN 100
80 POKESA+13,124:POKESA+15,165:POKESA+25,124:PO
  KESA+26,165
90 POKESA+39,205:POKESA+41,21:POKESA+123,205:POK
  ESA+124,189
100 POKESA+4,INT(SA/256):SYS SA:NEW
110 DATA 120,162,24,160,13,173,4,3,201,24,208,4
  ,162,13,160,67,142,4,3,140
120 DATA 5,3,88,96,32,13,67,152,72,169,0,141,0,
  255,133,176,133,180,166,22
130 DATA 164,23,134,167,132,168,170,189,0,2,240
  ,58,201,48,144,7,201,58,176
140 DATA 3,232,208,240,189,0,2,240,42,201,32,20
  8,4,164,180,240,31,201,34
150 DATA 208,6,165,180,73,1,133,180,230,176,164
  ,176,165,167,24,125,0,2,133
160 DATA 167,165,168,105,0,133,168,136,208,239,
  232,208,209,169,42,32,210
170 DATA 255,165,167,69,168,170,169,0,32,50,142
  ,169,32,32,210,255,32,210
180 DATA 255,169,13,32,210,255,104,168,96,104,1
  70,24,32,240,255,104,168
190 DATA 96,56,32,240,255,138,72,152,72,24,162,
  0,160,0,32,240,255,169
200 DATA 42,208,198
  
```

# AUTHORS WANTED!

*RUN* IS ALWAYS on the lookout for programs and articles that contain interesting and useful ideas. For the most part, those ideas come from you, our readers. We rely on you to keep our files well stocked with articles and programs from which to choose.

What kinds of articles do we need? We are looking for programs—of all kinds, shapes, sizes and colors. We need useful applications for the home, small business and school. We need utilities, programmers aids, creativity software and games.

We are sure many of you have developed unique programs that you use every day. You may not realize that a whole community of users is waiting to read about and share your creations.

If you are not a programmer, don't despair. We still need you. The introduction of new Commodore products—GEOS, the 1351 mouse, the 17xx series of RAM expanders and the 1581 drive—has opened up a vast area of topics for you to write about. What commercial software packages do you use that support these devices? What are their strengths and weaknesses? Users and potential users need to know.

These are just suggestions; we're sure you can think of more. Consider this an invitation to share your knowledge and computing experiences with tens of thousands of other Commodore users. And you will be rewarded for your efforts.

To help you submit those articles and programs for publication, we provide the *RUN* author's guidelines. These information sheets give you an idea of what kinds of material we are looking for and take you step by step through the process of preparing your articles for submission.

For a free copy, send a self-addressed, stamped, business-size envelope to:

Author Guidelines  
*RUN* Magazine  
80 Elm Street  
Peterborough, NH 03458

## LIST OF ADVERTISERS

(603) 924-7138 or (800) 441-4403

NATIONAL ADVERTISING SALES MANAGER: KEN BLAKEMAN

NORTHEAST SALES: BARBARA HOY

MIDWEST/SOUTHEAST SALES: NANCY POTTER-THOMPSON

WESTERN STATES SALES MANAGER: GIORGIO SALUTI, (415) 328-3470

Reader Service	Page	Reader Service	Page
412 ALX Digital	43	419 Precision, Inc.	1
404 Absolute Entertainment	CIII	421 Renco Computer	87
405 Absolute Entertainment	5	* <i>RUN</i>	
* Brantford Educational Services	11	Contest	2
420 Brown Boxes, Inc.	85	ReRUN Subscription	25
418 Creative Micro Design	77	RUN Works	42
402 Digital Solutions	CIV	GEOS Power Pak	45
411 Entertainment On-Line	13	Special Issue Disk	49
409 H&P Computers	37	* SOGWAP Software	85
* ICR Future Soft	16-18	417 Softech Computer Systems	93
413 Jason Ranheim	67	* Tab Books, Inc.	35
407 Joker Software Int'l.	CII	416 Tektonics Plus, Inc.	67
410 Lance Haffner Games	34	422 Top-Tech, Int'l.	89
414 Master Software	77	415 Ultrabyte	93
408 MicroLeague Sports	19	401 Xetec, Inc.	26

For further information from our advertisers, circle the corresponding Reader Service number on the adjoining card.

\*This advertiser prefers to be contacted directly.

This index is provided as an additional service. The publisher does not assume any liability for errors or omissions.

**RUN ALERT:** As a service to its readers, *RUN* will periodically publish the names of companies who are having difficulties meeting their customer obligations or who have gone out of business. Readers are advised to contact Susan Maizel, Customer Service Representative, *RUN* Magazine, 80 Elm St., Peterborough, NH 03458, before dealing with these companies: S&S Wholesalers, Compumed, Pro-Tech-Tronics, White House Computer, Prism Software (Waco, Texas), Underware and Starflite.

PRESIDENT  
MICHAEL PERLIS

VICE-PRESIDENT/GENERAL MANAGER  
ROGER MURPHY

VICE PRESIDENT  
STEPHEN TWOMBLY

CORPORATE CIRCULATION DIRECTOR: FRANK S. SMITH  
SINGLE COPY SALES MANAGER: LINDA RUTH

DIRECT SALES MANAGER: MICHAEL CARROLL

NEWSSTAND PROMOTION MANAGER: DEBBIE WALSH

DIRECTOR OF CREDIT SALES & COLLECTIONS: WILLIAM M. BOYER

CORPORATE PRODUCTION DIRECTOR: DENNIS CHRISTENSEN

CORPORATE PRODUCTION MANAGER: SUSAN GROSS; MANUFACTURING MANAGER: LYNN LAGASSE

TYPESETTING MANAGER: LINDA PALMISANO; SYSTEM SUPERVISOR: DOREEN MEANS

TYPESETTER: DEBRA A. DAVIES

**Manuscripts:** All manuscript contributions, queries, requests for writer's guidelines and any other editorial correspondence should be directed to *RUN*, Editorial Offices, 80 Elm St., Peterborough, NH 03458; telephone: 603-924-9471.

**Subscription problems or address changes:** Call 1-800-525-0643 (in Colorado, call 447-9330), or write to *RUN*, Subscription Services, PO Box 58711, Boulder, CO 80322-8711.

**Problems with advertisers:** Send a description of the problem and your current address to: *RUN*, 80 Elm Street, Peterborough, NH 03458, ATTN: Susan Maizel, Customer Service.

**Back Issues:** *RUN* back issues are available for \$3.50, plus \$1 postage and handling, from: *RUN*, Back Issue Orders, 80 Elm St., Peterborough, NH 03458.

**Problems with ReRUN:** Write to ReRUN, 80 Elm St., Peterborough, NH 03458, or call 1-800-343-0728.

**RUN's BBS:** The RUNning Board is *RUN*'s reader feedback bulletin board, which you can call anytime, day or night, seven days a week, for up-to-date information about the magazine, the Commodore industry and news and information of interest to all Commodore users. Call: 603-924-9704.

# Pilots needed for Interstellar Duty!

Garry Kitchen's

# Star Fighter™

**Coming Soon!**

Deep space missions available. Interested parties should contact:  
Recruiting Officer, Absolute Entertainment, Inc., PO Box 116,  
Glen Rock, NJ 07452.



Absolute Entertainment is a registered trademark and Garry Kitchen's Star Fighter is a trademark of Absolute Entertainment, Inc. © 1988 Absolute Entertainment, Inc.

Circle 404 on Reader Service card.

# THE ULTIMATE Word Processor from Digital Solutions

Here it is ...the ultimate in power ...the ultimate in ease of use...the ultimate in speed for your Commodore 64 & 128. All of the on-screen features that made Pocket Writer a 500,000 seller, plus these new exciting additions:

- Multiple columns. Print up to 4 newspaper-style columns.
- Macro capability. Record and invoke.
- Undo. To cancel last command.
- Markers. Mark up to 10 locations in one text.
- Book paging, odd-even and left/right.
- Line and box drawing modes.
- Word, sentence and paragraph count.
- Find/replace in either direction.
- Cursor movement by sentence and paragraph.
- Spelling Checker incorporated into both 64 & 128 programs. (Original Pocket Dictionary compatible)
- Automatic configuration for screen color, format and printer selection.
- On-screen text formatting and wordwrap. What you see is what you get!

#### *In 128 version only*

- Supports RAM expander and Mouse
- 25 or 50 lines of text displayed on screen
- Allows two files in memory at once for fast editing
- On-screen text enhancement including bold-face, underlines and *italics* (superscripts and subscripts)

*Plus all the features you've become accustomed to:*

- No complicated format commands embedded in text.
- Reads files generated by Pocket Filer 2 and Pocket Planner 2.



## Pocket Writer™ 3

Professional Word Processor  
for the Commodore 128 and 64

### Special Offer!

If you order your Pocket Writer™ 3 Program from Digital Solutions Inc. before January 31, 1989, you will receive a Free Pocket Writer™ Template Disk (retail value 24.95 U.S. (29.95 Can.))

**Don't wait! Order today!** You can get the Ultimate Word Processor from Digital Solutions. Send in the Order Form or call 416-731-8878 or Fax 416-731-8915 (credit cards orders only) Price is \$69.95 U.S. (\$79.95 Can.) We pay all shipping and handling charges.

Ontario residents add 8% Provincial Sales Tax of \$6.40 (Total \$86.35)

**Please check the version you require**     64     128

Name \_\_\_\_\_

Address \_\_\_\_\_ City \_\_\_\_\_

State/Prov. \_\_\_\_\_ Postal Code \_\_\_\_\_

Payment enclosed     Amex     Visa     Mastercard

Acct. \_\_\_\_\_ Exp. \_\_\_\_\_ Signature \_\_\_\_\_

 Digital  
Solutions  
Inc.

P.O. Box 345,  
Station A,  
Willowdale, Ontario,  
Canada M2N 5S9  
Phone: 416-731-8775  
Fax: 416-731-8915  
Circle 402 on Reader Service card.

Credit card orders must be signed.